ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

ANÁLISIS DEL COMPORTAMIENTO DE CONTROLADORES vRAN EN REDES 5G

(Performance of vRAN controllers in 5G networks)

Para acceder al Titulo de

Graduado en Ingeniería de Tecnologías de Telecomunicación

Autor: Cristina Aurora Hervella Baturone

Julio - 2020

E.T.S. DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACION

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

CALIFICACIÓN DEL TRABAJO FIN DE GRADO

Realizado por: Cristina Aurora Hervella Baturone
Director del TFG: Luis Francisco Diez Fernández y Ramón Agüero Calvo
Título: "Análisis del comportamiento de controladores vRAN en redes 5G"
Title: "Performance of vRAN controllers in 5G networks"

Presentado a examen el día: 14 de julio de 2020

para acceder al Título de

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Composición del Tribunal:

Presidente (Apellidos, Nombre):	Luis Muñoz Gutiérrez
Secretario (Apellidos, Nombre):	Ramón Agüero Calvo
Vocal (Apellidos, Nombre):	Víctor Manuel Fernández Solorzano

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente

Fdo.: El Secretario

Fdo.: El Vocal

Fdo.: El Director del TFG
(sólo si es distinto del Secretario)

Vº Bº del Subdirector

Trabajo Fin de Grado Nº
(a asignar por Secretaría)

Resumen

En la actualidad, la virtualización de las funciones de red de acceso radio, como papel fundamental en las redes 5G está ampliamente aceptado. Gracias a esta nueva configuración, se logran cumplir los exigentes requisitos respecto a retardo o tráfico, impuestos por el número masivo de usuarios que crece exponencialmente. Pese a que en un principio se propusieron enfoques totalmente centralizados, como los que ofrecían arquitecturas vRAN o C-RAN, resultaban no ser del todo viables debido a la sobrecarga de datos en los enlaces entre las entidades. En consecuencia, se proponen soluciones basadas en el *split* funcional flexible, donde un controlador central adapta el nivel de centralización en función de las circunstancias. Este trabajo se sitúa en este punto, realizando un simulador capaz de validar el nuevo modelo controlador, en desarrollo por el Grupo de Ingeniería Telemática. Esta solución surge ante la falta de trabajos que busquen modelar y analizar el rendimiento de dichas arquitecturas, en contraposición de los estudios existentes sobre su implementación. Se pretende, por ende, corroborar el funcionamiento correcto del simulador mediante modelos clásicos, como el M/M/1 o M/G/1 y, tras ello, analizar el rendimiento del controlador central, dependiendo de la política aplicada –tanto en sistemas con o sin pérdida de paquetes– comparándolo con el modelo del controlador vRAN para su validación.

Abstract

Nowadays, it is broadly accepted that Radio Access Network functions virtualization will play a key role to meet the stringent requirements of 5G networks. Thanks to this new configuration, it is possible to meet the strongly demanding requirements regarding delay times or traffic, imposed by the exponentially growing number of users. Although initially fully centralized approaches were proposed, they may impose unaffordable requirements over fronthaul links. Consequently, flexible functional split solutions are being proposed, where a central controller adapts the centralization level to the current circumstances. This work is located at this point, developing an event-driven simulator capable of validating a novel controller model, being developed by the Grupo de Ingeniería Telemática. In spite of the growing interest in this type of solutions, most of the existing works focus on real implementation, while little attention has been paid so far to performance modeling. Therefore, the aim is to corroborate the correct operation of the simulator, by means of classic models such as M/M/1 or M/G/1 and, after that, to analyze the performance of the vRAN controller, depending on the policy applied –both in systems with or without packet losses– comparing the results with those obteined by the controller model.

Índice general

Índice de figuras

Índice de tablas

Glosario

BBU BaseBand Unit.

BS Base Station.

BSC Base Station Controller.

BSS Base Station Subsystem.

BTS Base Transceiver Station.

C-RAN Cloud Radio Access Network.

CU Central Unit.

DU Distributed Unit.

eNB evolved Node B.

EPC Evolved Packet Core.

E-UTRAN Evolved UMTS Terrestrial Radio Access Network.

FDD Frequency Division Duplex.

FIERCE Future Internet Enabled Resilient smart CitiEs.

FIFO First In First Out.

GSM Global System for Mobile communications.

IoT Internet of Things.

LTE Long Term Evolution.

M2M Machine to Machine.

MS Mobile Station.

NFV Network Function Virtualization.

NGFI Next Generation Fronthaul Interface.

NSS Network and Switching Subsystem.

OFDMA Orthogonal Frequency Division Multiple Access.

OSS Operation Support Subsystem.

PK Pollaczek-Khintchine.

QoS Quality of Service.

RF Radio Frecuencia.

RNC Radio Network Component.

RRH Remote Radio Head.

SDN Software Defined Networking.

SF Spreading Factor.

TDD Time Division Duplex.

TDMA Time Division Multiple Access.

UE User Equipment.

UMTS Universal Mobile Telecommunications System.

vRAN Virtual Radio Access Network.

W-CDMA Wideband Code Division Multiple Access.

WRC'92 World Radiocommunications Conference 1992.

Capítulo 1

Introducción

Actualmente las redes de quinta generación o 5G están en auge y en proceso de implementarse. Este salto se ha podido realizar entre otros factores gracias a la utilización de un nuevo paradigma llamado Network Function Virtualization (NFV), siendo este posible por la explotación de las técnicas Software Defined Networking (SDN). El objetivo es lograr redes cada vez más virtuales, ya que de esta manera se consiguen tiempos de retardo menores e incluso respuestas en "tiempo real". Además, al estar los recursos en la nube, éstos son fáciles de compartir entre diferentes sistemas, permitiendo así un incremento de la eficiencia de los mismos. Para ello se necesitan potentes máquinas virtuales que sean capaces de soportar todo el tráfico, que crece de manera exponencial a causa del incremento de usuarios, los servicios Machine to Machine (M2M) o el Internet of Things (IoT). Otra característica a tener en cuenta es el retorno de una solución centralizada. En 4G se propuso una descentralización de los sistemas, cuyo resultado fue una limitación de usuarios y conexiones. En 5G se quiere superar ese problema, con una nueva entidad centralizada llamada Central Unit (CU) capaz de coordinar a las Distributed Unit (DU), conformando el conjunto de ambas la Base Station (BS).

La arquitectura que ofrece esta nueva solución pertenece a la familia de las Virtual Radio Access Network (vRAN). En concreto, en este trabajo se hablará del Cloud Radio Access Network (C-RAN). Esta solución implica diferentes Remote Radio Head (RRH) a las cuales los usuarios deben conectarse y que a su vez son controladas por BaseBand Unit (BBU) Virtuales. Estas entidades cubren diferentes funcionalidades de red: mientras que las más tradicionales pueden virtualizarse o centralizarse, las sobrantes deben permanecer a escasos metros de la antena. Con esta arquitectura se logra que el procesado de la información estuviese correctamente coordinado, pero también aparecen una serie de deficiencias. La comunicación entre las entidades de la BS tenía que soportar cargas de datos tan elevadas que podían no ser admisibles, ya que la RRH sólo realizaba operaciones básicas y el peso de las ellas recaía en la BBU. Por este motivo se optó por soluciones que separaban las funcionalidades de red, conocido como el *split* funcional. La clave para ello es encontrar cuál es la división de funcionalidades más eficiente, y es por este mismo motivo que se intenta encontrar soluciones más flexibles, las cuales permitan la variabilidad de las funciones dependiendo del servicio.

Este trabajo se centra en este último caso de *split* funcional flexible, sin profundizar en cómo hacer esta separación, sino en analizar el rendimiento del controlador central dependiendo de la política aplicada en cada caso. Para ello se realizará un simulador el cual se apoya en un modelo basado en la teoría de colas de Markov que se encuentra en desarrollo.

1.1. Objetivos

Se pretende analizar el comportamiento de controladores vRAN en redes 5G, por lo que se profundizará en la arquitectura C-RAN. En concreto se estudiará la posibilidad de modificar de manera dinámica el *split* funcional, lo que modificará la velocidad a la que se procesan las tramas (en base al número de funciones ejecutadas en el controlador). Se considerarán dos configuraciones: (1) con *buffer* de capacidad infinita, en el que no se producirán pérdidas; y (2) con *buffer* de capacidad finita, en el que se producirán pérdidas. Se analizarán parámetros de rendimiento con impacto en las comunicaciones sobre redes 5G, que imponen unos requisitos estrictos en cuanto a retardo: tiempo de permanencia en el controlador, y probabilidad de pérdida, analizando la relación entre ellos.

Para el estudio, se desarrollará un simulador por eventos en el lenguaje de programación C++, que se utilizará para llevar a cabo los experimentos correspondientes. La validez de la implementación se corroborará analizando el comportamiento de modelos clásicos de teoría de tráfico como el modelo M/M/1 o el M/G/1. Además, el rendimiento del controlador (obtenido a partir del simulador) se comparará con un modelo teórico, basado en Cadenas de Markov, en desarrollo por el Grupo de Ingeniería Telemática.

1.2. Estructura

El trabajo presentado, tras haberlo contextualizado e identificando sus objetivos se quieren cumplir, sigue la siguiente estructura. En el Capítulo 2 se hablarán de los fundamentos teóricos en los que se ha apoyado, es decir, se explicará cómo se ha llegado a la tecnología en la que se basa este trabajo –tanto la evolución de las redes móviles hasta el 5G, como la arquitectura conocida como C-RAN o Cloud-RAN– y los diferentes modelos de teletráfico (o dimensionado) en los que se apoya el simulador. En el Capítulo 3 se explicará la configuración y la implementación elegida para el simulador en los diferentes escenarios seleccionados, empezando por los modelos clásicos anteriormente explicados (M/M/1 y M/G/1) seguido por el modelo teórico del controlador. En el Capítulo 4 se exponen los resultados obtenidos del simulador en diferentes configuraciones, así cómo las comparativas de este con los modelos teóricos comentados. Por ende, se analizará su funcionamiento y, tras ello, se valida el modelo controlador. Por último, el Capítulo 5 presenta la conclusión a la que se ha llegado tras los resultados obtenidos, de contribuciones realizadas a partir de este proyecto, y de las diferentes líneas de trabajo que pueden surgir a la finalización del mismo.

Capítulo 2

Fundamentos teóricos

Para contextualizar el marco en el que se sitúa el trabajo, en este capítulo se analizará la evolución de las redes móviles centrándonos en su arquitectura, sobre todo en la quinta generación. Tras ello, el siguiente punto será la explicación de la arquitectura C-RAN y el *split* funcional. Por último se comentará en qué modelos clásicos del dimensionado de redes se ha apoyado el desarrollo del simulador.

2.1. Evolución de la arquitectura de redes móviles

A día de hoy se habla de la quinta generación de redes móviles, el 5G. Pero para saber cómo se ha llegado a ella, es importante comprender la evolución por todas las generaciones anteriores. Para este trabajo, lo central es incidir en su arquitectura.

Primera y Segunda Generación

La Primera Generación de redes celulares (1G) ofrecía una buena calidad de voz pero tenía una eficiencia espectral bastante limitada, una vida útil corta y de coste elevado. La siguiente generación, el 2G o Segunda Generación, tuvo que enfrentarse a estos problemas y para solventarlos apareció entonces el estándar GSM [1].

Esta tecnología gestiona el acceso múltiple con Time Division Multiple Access (TDMA), utilizando un esquema de multiplexado Frequency Division Duplex (FDD). Esto implica que el ancho de banda total disponible sea de 25 MHz, el cual se divide en las 125 tramas (124 portadoras más 1 de control) que están espaciadas 200 KHz entre sí. Las tramas a su vez tienen 8 *slots* o canales, resultando un total de 1000, siendo 8 de ellas de control [2]. Su arquitectura se compone de cuatro elementos principales: Mobile Station (MS), Base Station Subsystem (BSS), Network and Switching Subsystem (NSS) y Operation Support Subsystem (OSS).

Lo más notorio en el ámbito del trabajo y en lo que se centrará esta sección, es la separación en el Subsistema de Estación Base (BSS) del Base Transceiver Station (BTS) y Base Station Controller (BSC). La primera se encarga de la recepción, transmisión y procesado de señal que llega de los propios usuarios, en cambio la segunda se ocupa de las conexiones con otros elementos y de las operaciones necesarias para la asignación de canales móviles o control de potencia.

Por tanto, la línea de trabajo sería la siguiente: los usuarios pueden conectarse a una serie de BTS's, a las que envían su información; éstas, a su vez, dirigen esta información hasta una BSC, la cual se encarga del procesado de la misma.

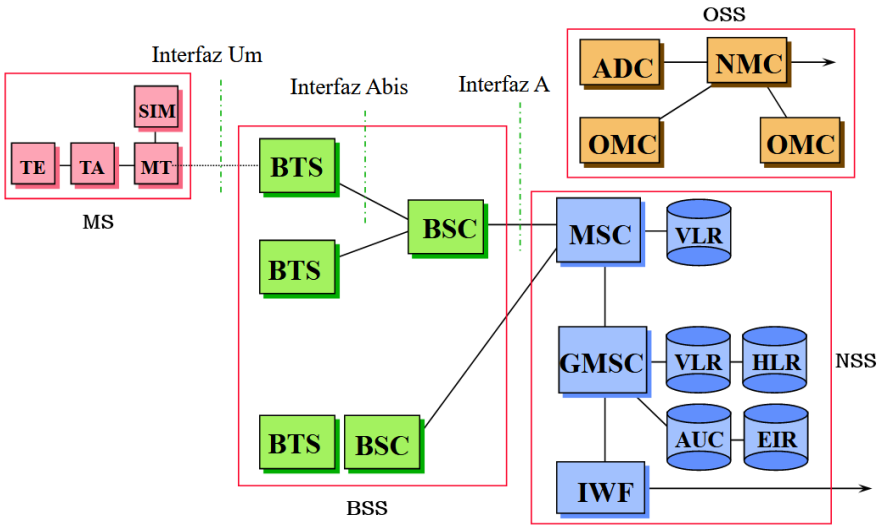

Figura 2.1: Estructura de GSM [2]

Como se puede ver en la Figura 2.1, además de los elementos anteriormente comentados, se observa el sistema centralizado que sigue la tecnología, en el que una BSC da servicio a varias BTSs, gracias a su gran capacidad de cómputo, de la cual carecen sus "hijas". Este hecho genera una serie de ventajas, como la obtención de BTSs a un coste reducido o la cooperación entre ellas para conseguir una mejora en el rendimiento.

Tercera Generación

Para la Tercera Generación de redes celulares (3G) se quiso desarrollar un nuevo estándar que fuera universal, con tasas binarias más elevadas, con una mejora de seguridad y soporte de servicios multimedia [3]. De ahí nació Universal Mobile Telecommunications System (UMTS), tras pasar por un largo proceso de estandarización y llegar hasta un consenso. Se basa en el empleo de una interfaz radio Wideband Code Division Multiple Access (W-CDMA) que opera con dos modos, FDD y Time Division Duplex (TDD). Las bandas de frecuencia para cada modo son distintas. Mientras que en FDD utiliza las bandas de 1920-1980 MHz en sentido ascendente y de 2110-2170 MHz en el descendente, en TDD ocupa 1as bandas de 1900-1920 MHz y 2010-2025 MHz, en sentido ascendente y descendente, respectivamente [5]. Igualmente hay que remarcar que esto solo es aplicable en Europa, ya que el espectro varía en diferentes regiones como se puede ver en la Figura 2.2. Todas estas características permiten un ancho de banda que alcanza los 2Mbps en instalaciones estacionarias, además de una mayor eficiencia espectral respecto a GSM.

Otros parámetros a tener en cuenta en comparativa con GSM son los relacionados con la estructura de las tramas. Cada trama tiene una longitud de 10 milisegundos y está compuesta de 15 *slots* de 2560 bits [6]. El espaciado entre las portadoras es de 5 MHz, con una tolerancia de 200kHz. El resultado final es una velocidad de trama de 3.84 Mchips por

segundo, la cual está relacionada con la tasa de bit gracias al factor de ensanchado, Spreading Factor (SF), cuyo su rango varía desde 4 a 256. A mayor SF, menor es la tasa de bit [7].

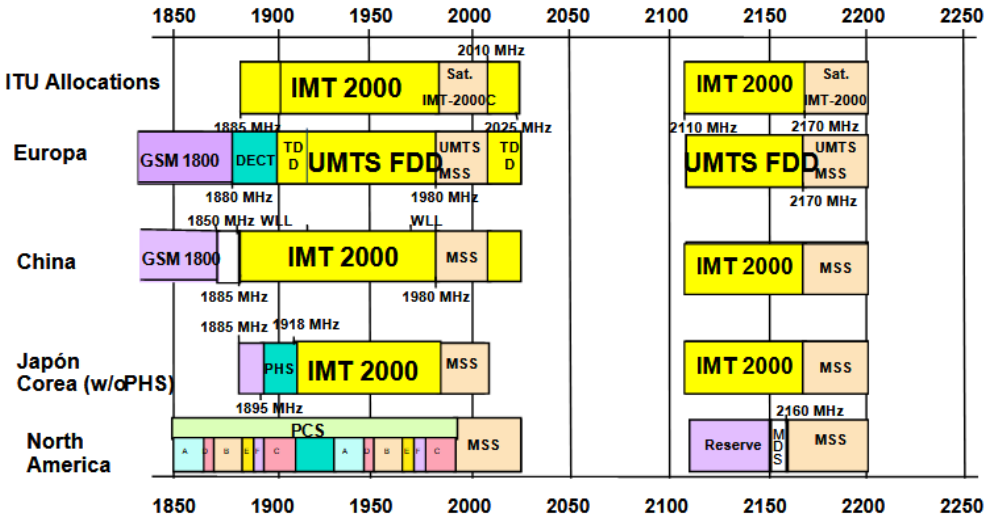

Figura 2.2: Espectro identificado en WRC'92 [3]

Tras haber mostrado cómo es UMTS, se entra en su arquitectura, ya que es lo que mayor interés tiene en el marco de este trabajo. En contraposición a las diferencias que se han visto hasta ahora con GSM, ésta no varía en exceso.

Como se puede ver en la Figura 2.3, se sigue manteniendo la estructura centralizada. Además, el usuario es ahora denominado User Equipment (UE) o móvil, y no se conecta a BTS, sino a una estación llamada Node B. Este elemento se encuentra en la capa física de la interfaz, realizando operaciones propias de este nivel, por lo que se tiene un grado mayor de sencillez. El siguiente elemento se correspondería con el Radio Network Component (RNC), siendo su equivalente en GSM el BSC. Su operación se basa en controlar uno o varios Nodos B, y gestionar los recursos a un nivel superior [8].

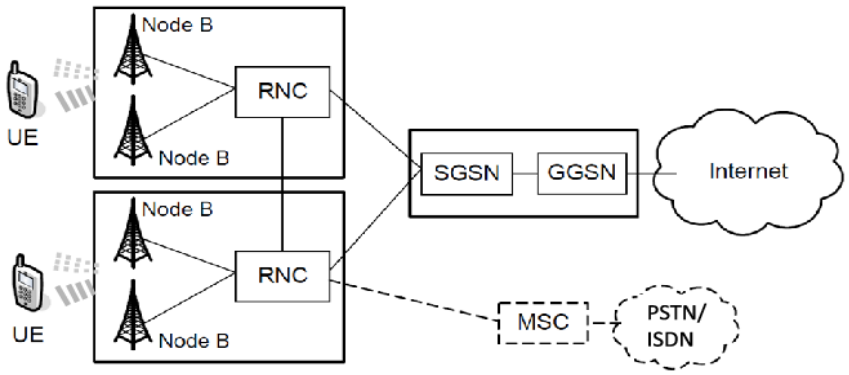

Figura 2.3: Arquitectura de UMTS [4]

Cuarta Generación

La siguiente Generación, llamada 4G comúnmente, se diseñó ante las carencias de 3G de limitación del ancho de banda y ante el avance tecnológico en el ámbito inalámbrico. Se creó para unir diferentes tecnologías no cableadas, de tal manera que proporcionen acceso de banda ancha e itinerancia mundial (roaming) [10].

El estándar tras el 4G es Long Term Evolution (LTE), cuyo objetivo principal es conseguir que todos los servicios sean soportados por el protocolo IP. De ahí que también se le conozca a esta generación como "all-IP" [11]. Como se puede ver en la Figura 2.4, se compone de tres elementos principales: el UE como en UMTS, la red de acceso Evolved UMTS Terrestrial Radio Access Network (E-UTRAN) y la red troncal Evolved Packet Core (EPC).

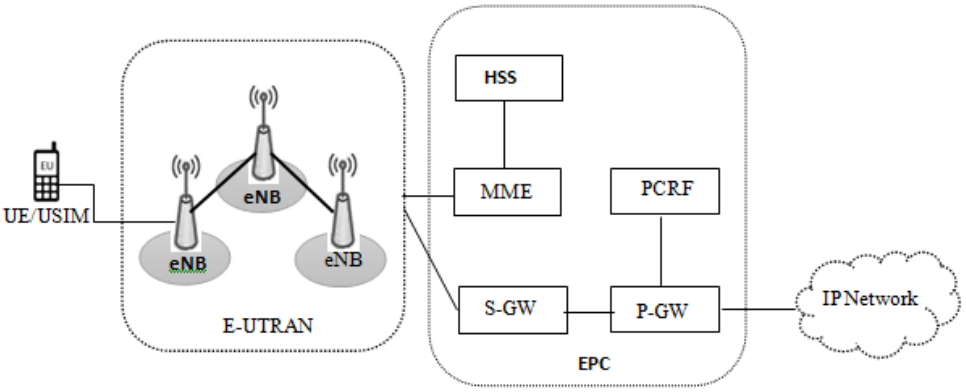

Figura 2.4: Arquitectura de LTE [9]

En E-UTRAN se puede observar un cambio notable respecto a las generaciones anteriores, ya que antes se tenía una BS (BTS o Nodo B) y un equipo controlador (BSC o RNC) mientras que aquí hay una única entidad de red, el evolved Node B (eNB). La funcionalidad de eNB es proporcionar conectividad entre los UE y EPC. Además, estas entidades pueden comunicarse entre sí para una gestión más eficiente de los recursos, así como del tráfico (en traspasos o handover) gracias a la interfaz X2. El EPC es responsable de proporcionar la conectividad IP.

Por otro lado, este sistema presenta ciertos inconvenientes, como su escasa escalabilidad o la limitada cooperación entre los eNBs. La tecnolgía de acceso evoluciona a Orthogonal Frequency Division Multiple Access (OFDMA). Esta tecnología multiportadora hace posible la subdivisión del ancho de banda disponible en una multitud de subportadoras de banda estrecha ortogonales entre sí, que se pueden compartir (o no) entre diferentes usuarios, dotando así la calidad de servicio deseada para cada caso [12]. Las bandas de frecuencia en LTE van desde los 700 MHz hasta los 2.7 GHz y están caracterizadas por su flexibilidad, ya que las subportadoras se pueden adaptar en tiempo real, dependiendo de las condiciones del canal o de la demanda.

Quinta Generación

Para este año se esperaba que la Quinta Generación o 5G hubiera estado presente alrededor del mundo [13]. En cambio, en la actualidad, esta nueva tecnología se presenta en algunas regiones o países y aún le queda camino por recorrer. El 5G supone un reto respecto a las generaciones anteriores, pues pretende una red inalámbrica capaz de hacer frente a los seis desafíos impuestos por la generación anterior [14]:

1. Mayor capacidad (1000 veces superior)
2. Mayor velocidad de datos (entre 10 y 100 veces superior)
3. Menor latencia de extremo a extremo (inferior a 5 milisegundos)
4. Conectividad masiva de dispositivos (entre 10 y 100 veces superior)
5. Abastecimiento constante de calidad de experiencia.
6. Coste sostenible.

Estos requisitos tan exigentes son necesarios debido a comunicaciones M2M y el IoT, que provocan un incremento exponencial de usuarios. Además, en el mundo actual todos los dispositivos tienen conectividad a Internet. Todo ello provoca un ancho de banda insuficiente para las conexiones simultáneas que se pueden dar. Otro de los motivos es cumplir los retos del IoT, como enviar información de cualquier índole en dónde sea y cuándo se quiera. Uno de los mayores y más importantes desafíos es conseguir una latencia inferior a 5 milisegundos, lo que daría lugar a unas comunicaciones en, prácticamente, tiempo real. Para terminar, sabiendo que se tratará con más detenimiento en la siguiente sección, es la arquitectura; en este caso, se llega a un consenso entre un sistema centralizado como el que se encontraba en 2G/3G y otro más distribuido como el de 4G.

2.2. Cloud-RAN y *split* funcional

La evolución de la arquitectura de la red de acceso a lo largo de las distintas generaciones ha llevado al 5G a la centralización de las funciones de red. Esto se logra gracias a la explotación de las técnicas SDN, las cuales llevan al paradigma NFV. En este nuevo modelo, las funciones tradicionales de las BSs son tanto virtualizadas como centralizadas, mientras que las funciones restantes permanecen más cercanas a la antena. Se obtiene entonces una división en la BS: la CU y la DU, donde la primera gestiona la segunda.

El proceso para la conexión al sistema sería el siguiente: los usuarios se conectan mediante una unidad llamada RRH y esta a su vez, es gestionada y conectada por BBU Virtualizadas (correspondiéndose al DU y al CU, respectivamente). El enlace entre ambas terminales sería el *fronthaul* [15]. Se le debe dar cierta importancia a este elemento, ya que sufre un cambio notable con la generación anterior, pasando de tener una longitud de centímetros o metros a ser de kilómetros. También hay que prestar atención a su capacidad, ya que ahora debe incrementarse unas 15 veces respecto a la del *backhaul*, enlace que conecta las BBUs con el núcleo de

red, debido al incremento de volumen de datos por usuario. Es por ello que se potencia el uso de fibra óptica como tecnología de acceso.

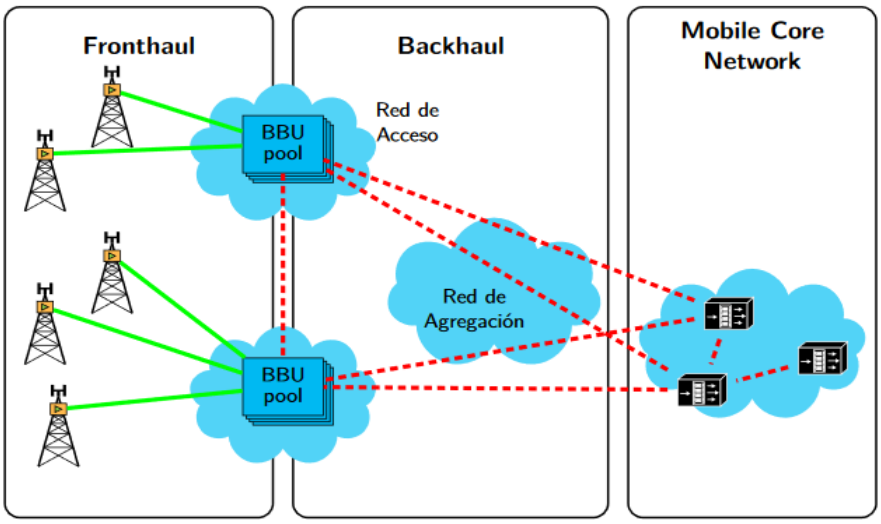

Figura 2.5: Arquitectura de C-RAN [16]

El resultado se puede observar en la Figura 2.5; se busca una nueva arquitectura de acceso a red dentro de la familia de las vRAN, llamada C-RAN [17] [18]. Esta arquitectura hace frente a los nuevos requisitos de capacidad, cobertura y experiencia del usuario, y es la combinación de tres factores: virtualización, centralización de red y coordinación de los nodos. Debido a esta segunda característica, la letra "C" se podrá también interpretar como *Centralized*, aunque se hará referencia a *Cloud*, como ya se ha visto.

Split funcional

En el modelo C-RAN se requiere que el RRH realice únicamente funciones básicas de Radio Frecuencia (RF). Sin embargo, este enfoque exige una alta capacidad de comunicación en el *fronthaul* entre el DU y el CU, lo cual puede no ser viable. De esta forma se vuelve a trabajar (desde el mundo académico hasta la industria u organismos de normalización) en definir soluciones que resuelvan esta limitación y que permitan la selección de diferentes *splits* en la BS [19].

Esta técnica del *split* funcional implica la división de las funcionalidades de red entre la BBU y las RRHs. Al virtualizar las funciones de red la decisión clave es averiguar qué separación funcional es la adecuada, es decir, qué funciones son necesarias en la CU y cuáles se pueden manejar en las DUs. Esta división tiene que cumplir unos requisitos, como es su capacidad de adaptarse o ajustarse a la red deseada, buscando siempre la manera más óptima de asegurar el retardo mínimo de las tramas, cumpliendo con las características de la red y del procesado requerido en ese instante. En la actualidad se encuentran posibles *splits* funcionales, definiendo dónde podrían situarse estas separaciones y las operaciones realizadas en ellas, como se muestran en los estudios [20] [21].

Adicionalmente, existe otra solución en la que las propias funciones varían dinámicamente [22] [23]. Esto se conoce comúnmente como los *splits* funcionales flexibles, debido a su adaptabilidad a los requisitos de la red en determinados momentos. En este caso, un controlador central selecciona cuál es el nivel de virtualización de una o varias DUs, según las necesidades de un servicio. Este controlador a su vez, necesita gestionar los propios recursos de virtualización para que los *split* elegidos sean manejables. Se presenta un análisis en [24] sobre el impacto de los diferentes *splits* funcionales en el ámbito del Quality of Service (QoS), energía y coste.

2.3. Dimensionado de redes

El dimensionado de redes consiste en calcular la cantidad de recursos necesarios para satisfacer el tráfico de una serie de servicios en una red, siendo el tráfico el número medio de llamadas en curso de un sistema [25]. Es fundamental, ya que los operadores buscan ofrecer a sus clientes un servicio adecuado de manera rentable; dependiendo de la demanda se busca una capacidad más o menos elevada. Este dimensionado se lleva a cabo normalmente mediante modelos matemáticos como los que se van a mostrar a continuación, que varían en función de las necesidades de la propia red. En este capítulo se realizará una breve explicación sobre dos modelos clásicos: el sistema de espera puro M/M/1 y su generalización [26] [27].

Ambos modelos se rigen por la notación Kendall (A/B/C/D/E/F):

- A: Distribución de llegadas al sistema.
- B: Distribución del tiempo de servicio.
- C: Número de recursos disponibles.
- D: Número de clientes/servicios que puede haber en el sistema (en un recurso o en espera). En ninguno de los dos modelos se indica, así que se asume que es infinito y que se está en un sistema de espera pura, dicho de otra forma, de cola infinita y sin pérdidas.
- E: Número de fuentes. Como en el anterior, al no indicarse se entiende que es infinito.
- F: Disciplina de la cola. Al igual que en los anteriores, tampoco se muestra y se supone que ambos son First In First Out (FIFO).

2.3.1. M/M/1

El M/M/1 se trata de un sistema en el cual las llamadas siguen un proceso de Poisson, es decir, el tiempo entre llegadas consecutivas sigue una distribución exponencial negativa de tasa λ. Que el proceso sea de Poisson también implica que las llegadas sean sin memoria, en otras palabras, cada llegada es independiente de intervalos anteriores o futuros. La probabilidad de que haya k llegadas en T se calcula a partir de:

$$Pr\{k \text{ llegadas en T } \} = P_k(T) = \frac{(\lambda T)^k}{k!} e^{-\lambda T}$$

Además, la distribución del tiempo de servicio sigue una variable aleatoria exponencial, con media $1/\mu$. Por último, existe un único recurso que gestiona las peticiones. Se modela este sistema como una sucesión de estados que representan el número de paquetes en el mismo, tanto en el recurso (máximo de 1 en este caso) como esperando. La Figura 2.6 representa su diagrama de estados, en el que se puede observar cómo la tasa de nacimiento y la tasa de muerte, al existir solo un servidor, son constantes.

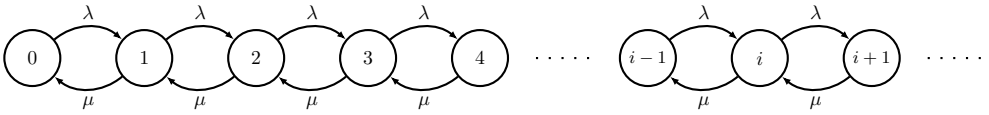

Figura 2.6: Diagrama de estados M/M/1

La probabilidad de estar en un estado se determina con la siguiente expresión matemática:

$$P_i = \rho^i(1-\rho)$$

Siendo $\rho = \lambda/\mu$ el tráfico o número medio de paquetes cursados en el sistema. También coincide con el grado de ocupación de los recursos.

A partir de estas expresiones se caracteriza el sistema, obteniendo el número medio de clientes en el sistema ($\overline{N_T}$), número medio de clientes en la cola ($\overline{N_Q}$) y el tiempo medio en el sistema y en la cola ($\overline{T_T}, \overline{T_Q}$) aplicando la relación de Little ($N = T\lambda$)

$$\overline{N_T} = \sum_{k=0}^{\infty} kP_k = \sum_{k=0}^{\infty} k\rho^k(1-\rho) = \rho(1-\rho)\sum_{k=0}^{\infty} k\rho^{k-1} = \frac{\rho}{1-\rho}$$

$$\overline{N_Q} = \sum_{k=1}^{\infty}(k-1)P_k = \sum_{k=1}^{\infty}(k-1)\rho^k(1-\rho) = \sum_{t=0}^{\infty} t\rho^{t+1}(1-\rho) = \frac{\rho^2}{1-\rho}$$

$$\left.\begin{aligned}\overline{T_T} &= \frac{\overline{N_T}}{\lambda} = \frac{\rho}{1-\rho}\frac{1}{\lambda} = \frac{1}{\mu-\lambda}\\ \overline{T_Q} &= \frac{\overline{N_Q}}{\lambda} = \frac{\rho^2}{1-\rho}\frac{1}{\lambda} = \frac{\rho}{\mu-\lambda}\end{aligned}\right\} \quad \overline{T_T} - \overline{T_Q} = \frac{1}{\mu-\lambda} - \frac{\rho}{\mu-\lambda} = \frac{1}{\mu} = T_s$$ [1]

2.3.2. M/G/1

El modelo M/G/1 simular al anterior, ya que se trata de un modelo genérico que engloba al M/M/1 y a otros, como el M/D/1, el M/Ek/1 y el M/Hk/1.

Todos comparten características similares: sus llegadas se regulan según un proceso de Poisson sin memoria, el tiempo de servicio es una variable aleatoria genérica (no necesariamente exponencial como el caso anterior); se tiene un único servidor; y no hay pérdida en el servicio.

Para llegar a un sistema que pueda englobar a los anteriores se usa la fórmula Pollaczek-Khintchine (PK), determinando el tiempo de espera promedio. Una de las variables de dicha fórmula es el coeficiente de

[1] Tiempo de servicio

variación ($C(T_s)$), la cual da información acerca de la dispersión relativa de, en este caso, el tiempo de servicio del sistema.

$$\overline{T_Q} = \frac{1 + C(T_s)^2}{2} \cdot \frac{\rho}{1 - \rho} \cdot E(T_s)$$

Esta fórmula se puede particularizar, dependiendo del tipo de variable que se use para el tiempo de servicio, como en el M/M/1, una exponencial negativa. Otro caso útil en este trabajo será dejar esta variable como una constante. Por ende, se obtienen dos expresiones matemáticas para obtener el tiempo medio en cola ($\overline{T_Q}$) y el número medio de clientes en cola ($\overline{N_Q}$).

$$\left.\begin{aligned} \overline{T_Q} &= \frac{1 + C(T_s)^2}{2} \cdot \frac{\rho}{1 - \rho} \cdot E(T_s) \\ \overline{N_Q} &= \frac{1 + C(T_s)^2}{2} \cdot \frac{\rho^2}{1 - \rho} \end{aligned}\right\} \quad \begin{aligned} &C(T_s) = 1 \text{ si la } va \text{ es exp. negativa} \\ &C(T_s) = 0 \text{ si la } va \text{ es constante} \end{aligned}$$

Capítulo 3

Modelado del sistema e implementación

Este trabajo, como ya se ha comentado, se centra en la utilización de un *split* funcional flexible. Uno de los desafíos principales de esta nueva arquitectura es la gestión del enlace *fronthaul*, ya que antes sólo era un conjunto de enlaces dedicados, y ahora es una red compleja basada en conmutación de paquetes. Es por este motivo que se ha empezado a definir la Next Generation Fronthaul Interface (NGFI), responsable de la gestión en la comunicación entre las entidades que componen una BS [28] [29]. Tras ello, queda encontrar un modelo adecuado de controladores que sea capaz de cumplir con los parámetros necesarios, y tener un buen rendimiento. Es aquí donde se sitúa el trabajo, diseñando un simulador que permita afrontar el análisis del rendimiento de la entidad controladora central, dependiendo de la política aplicada. Esta herramienta se ha particularizado en cuatro escenarios genéricos: (1) modelos clásicos como el M/M/1 y el M/G/1; (2) el modelo clásico M/M/1 truncado, es decir, adaptándolo para contemplar pérdidas; (3) el modelo controlador sin pérdidas; y (4) el modelo controlador con pérdidas. Mientras que los dos primeros sirven para validar el correcto funcionamiento del simulador, los dos últimos son los de mayor importancia, ya que ayudan a la validación del modelo controlador que posteriormente se explicará en detalle.

Las diferentes configuraciones del simulador siguen una estructura genérica la cual se basa en un elemento central que gestiona los distintos eventos que ocurren en el recurso, ya sean de entrada o salida de paquetes, como de otra índole (se ampliarán en el modelo no clásico). Al simulador hay que indicarle las características del sistema, como el valor de la tasa de llegadas (λ), el tiempo de servicio ($1/\mu$), etcétera. La disciplina de la cola para todos los escenarios vuelve a ser FIFO, es decir, el primero que entra a la cola, es el primero que sale del sistema. Otro elemento a tener en cuenta es el número de paquetes que se analizan, $numEntries$; se usarán unos 10.000 o 100.000 paquetes para conseguir resultados estadísticamente fiables. Se explicará a continuación cómo funciona en detalle el simulador dependiendo del escenario a tratar.

3.1. Modelos Clásicos

En esta sección se particulariza el simulador para analizar modelos teóricos clásicos. Además, como se requiere para el modelo controlador un sistema con pérdidas, se ajustará el M/M/1 y el simulador para ello. El motivo para realizar estos ejemplos es la validación del correcto funcionamiento del mismo, comparando con las fórmulas de la sección 2.3, lo cual se verá en el Capítulo 4.

3.1.1. M/M/1 y M/G/1

En el Capítulo 2 ya se explicaron los modelos clásicos M/M/1 y M/G/1 teóricamente, por lo que no será necesario profundizar sobre ellos.

El simulador para este caso es el más sencillo. Para su funcionamiento sólo hay que indicarle la tasa de llegadas (λ) y el tiempo de servicio (μ^{-1}). El simulador trabaja con dos eventos: entrada y salida. El primero indica que un paquete ha llegado al sistema, lo que lleva al siguiente paso lógico, saber si el recurso está ocupado o no. Si lo está, el paquete en cuestión se añade a la cola; si no, se genera un evento de salida para el mismo. Este segundo evento supone que el paquete sale del sistema, es decir, ha sido procesado con éxito. Además, mira si hay elementos en la cola y, si es así, los elimina de la misma y genera su correspondiente evento de salida. Si por el contrario el *buffer* (cola) está vacío, se indica que el recurso está libre.

Pseudocódigo

En el Algoritmo 1 se expone de manera más detallada el pseudocódigo del simulador que se acaba de explicar. Es importante aclarar que en todo el proceso los paquetes almacenan el instante en el que entran al sistema (cola), cuándo al servidor (recurso) y cuándo salen, ya que son datos vitales para poder realizar todos los cálculos precisos; esto se puede observar en las líneas 3, 12-18 y 12, respectivamente.

Algorithm 1 M/M/1 y M/G/1

Require: λ, μ

```
1: procedure SimulatorMM1
2: ProcessArrivalEvent:
3:     setPktTime (SYSTEM-IN, pkt, time)
4:     if (numPkts < numEntries) then
5:         generateEvent(ARRIVAL, time, nextPkt)
6:         numPkts++
7:     end if
8:     if (resourceBusy = true) then
9:         queuePush(pkt)
10:    else
11:        generateEvent(DEPARTURE, time, pkt)
12:        setPktTime (SERVER-IN, pkt, time)
13:    end if
14: ProcessDepartureEvent:
15:    setPktTime (SYSTEM-OUT, pkt, time)
16:    if (queue > 0) then
17:        generateEvent(DEPARTURE, time, nextPkt)
18:        setPktTime (SERVER-IN, nextPkt, time)
19:    else
20:        resourceBusy = false
21:    end if
22: end procedure
```

3.1.2. M/M/1 truncado: con *Buffer* de Espera Finito

En esta particularización del simulador se trabaja con un modelo M/M/1 modificado o truncado, un modelo que sigue la siguiente notación M/M/1/D. Se recuerda que "D" es el número de clientes que puede haber en el sistema –ver Sección 2.3. Este cambio implica que el sistema sea de espera finito, significando la posible pérdida de paquetes y una nueva variable llamada *buffer*. Este dato es quien indica al simulador cuántos paquetes pueden permanecer en la cola en el mismo instante como máximo, descartando[1] aquellos que lleguen cuando el sistema esté lleno. No es baladí ahora calcular la probabilidad de pérdida de los paquetes, cambiante según las características del sistema. Excepto esta modificación, el modelo en sí funciona igual que el anterior.

Respecto al modo de operación entre simulador previo y el actual no existe una variación notable. Se vuelve a tratar con eventos de entrada y de salida, donde el segundo actúa de la misma manera que en la configuración anterior. La diferencia se encuentra en cómo actúa en el evento de entrada, ya que no solo hay que tener en cuenta si el recurso está ocupado sino que también, en el supuesto que así sea, mirar cuántos clientes hay en cola. Si ese número coincide con el de *buffer*, el paquete será descartado.

[1] Se aclara que cuando se descartan paquetes significa que éstos han sido analizados, pero se han perdido por la falta de espacio en el sistema.

Pseudocódigo

Se expresa en el Algoritmo 2 el nuevo caso del simulador, siendo prácticamente igual al previo exceptuando el manejo de nuevos clientes si la cola se igualase al número máximo de usuarios soportados en el sistema.

Algorithm 2 M/M/1 truncado

Require: λ, μ, $buffer$

```
1: procedure SimulatorBufferMM1
2: ProcessArrivalEvent:
3:     setPktTime (SYSTEM-IN, pkt, time)
4:     if (numPkts < numEntries) then
5:         generateEvent(ARRIVAL, time, nextPkt)
6:         numPkts++
7:     end if
8:     if (resourceBusy = true) then
9:         if (queue = buffer) then
10:            lostPkt++
11:            break
12:        end if
13:        queuePush(pkt)
14:    else
15:        generateEvent(DEPARTURE, time, pkt)
16:        setPktTime (SERVER-IN, pkt, time)
17:    end if
18: ProcessDepartureEvent:
19:    setPktTime (SYSTEM-OUT, pkt, time)
20:    if (queue > 0) then
21:        generateEvent(DEPARTURE, time, nextPkt)
22:        setPktTime (SERVER-IN, nextPkt, time)
23:    else
24:        resourceBusy = false
25:    end if
26: end procedure
```

3.2. Modelo Controlador

Antes de seguir con el simulador, hay que realizar una pequeña explicación de cómo funciona el modelo controlador. Cabe señalar que este modelo propuesto por el Grupo de Ingeniería Telemática no tiene como objetivo proporcionar la separación de funcionalidades en los *splits* del algoritmo, sino predecir el rendimiento del controlador una vez que se aplica una política en particular. Para ello se basa en la teoría de las Cadenas de Markov, como se puede ver en el diagrama de estados representado en la Figura 3.1. En cada uno de los estados, simbolizados por bolas, se muestra cuántos paquetes hay en el sistema y su correspondiente *split* (β,s).

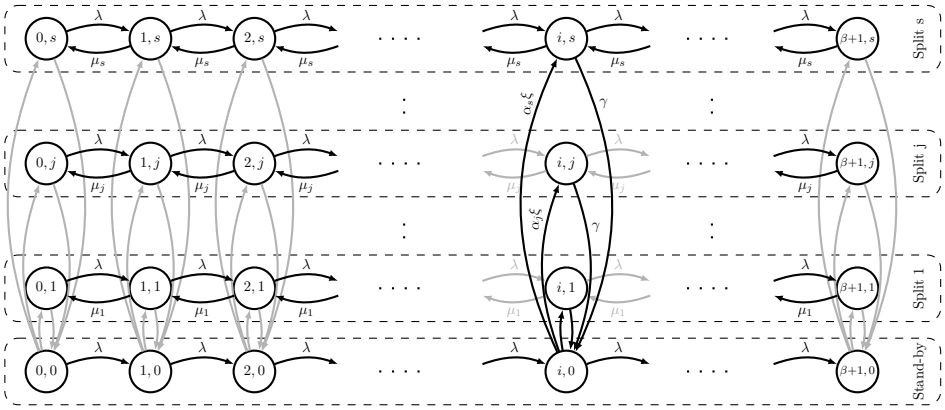

Figura 3.1: Diagrama de estados Modelo Controlador

En este diagrama se puede observar la existencia de una primera configuración, el *Stand-by*, y un número s de *splits*. La idea tras el mismo es la siguiente. Cada *split* funciona de manera casi independiente entre sí, teniendo tasas de servicio diferentes $(\mu_1, \mu_2, ..., \mu_s)$ pero, en cambio, compartiendo el valor medio del tiempo de llegadas (λ) y el *buffer* –ya sea infinito para el sistema sin pérdidas u otro a definir si las hay. Sólo puede haber en funcionamiento un único *split* por lo que surge entonces la duda de cómo se coordinan dichos elementos. Las transiciones entre ellos se llevan a cabo tras pasar por el modo *Stand-by*. Como se demuestra en el diagrama, en esta configuración el sistema queda suspendido, sin la posiblidad de procesar clientes; de manera que tanto los nuevos paquetes como los anteriores quedan en espera. Es de vital importancia ya que, para poder cambiar entre *splits* coordinadamente, se deberá pasar por esta configuración y después, elegir una nueva para poder procesar a los clientes con normalidad –valiéndose para ello de sus probabilidades de uso (α_i). Dependiendo del controlador, el tiempo cada cuánto este comprueba qué *split* es el siguiente que corresponde (γ^{-1}) y, además, el propio tiempo de espera en el modo de reposo (ξ^{-1}) puede variar.

Como ya se ha dicho, este modelado tiene como objetivo pronosticar el rendimiento del controlador con *splits* funcionales flexibles; este trabajo en cambio, desea analizar dicho rendimiento para poder validar el propio modelo. A continuación se muestran dos ejemplos en los que se explicarán y cómo se ha desarrollado el simulador para cada uno.

3.2.1. Con *Buffer* de Espera Infinito

El primer caso de este modelo controlador se basa en un sistema con espera infinita, es decir, un sistema sin pérdidas en el que el *buffer* tiende a infinito. El simulador parte del primer ejemplo realizado con una serie de cambios, donde se deben indicar un conjunto de datos como la tasa de llegadas (λ), la tasa de servicio para cada *split* $(\mu_1, \mu_2, ..., \mu_s)$, el número de ellos (s) y la probabilidad de los mismos (α_i). Otro factor es el cambio de *split*, el cual se realiza tras un cierto intervalo de tiempo, modelado por una variable aleatoria exponencial de tasa γ; el tiempo de espera en este estado de reposo también se modela con una variable de la misma familia, con valor medio ξ^{-1}.

Una novedad respecto a las configuraciones previas es la existencia de dos nuevos eventos: parar el sistema y restablecerlo. Los otros eventos, de entrada y salida de paquetes, no será necesario explicarlos, ya que funcionan de la misma forma –obviando el caso en el que el sistema esté en reposo y lleguen entradas, en cuyo momento se almacenarán directamente en la cola sin mirar si el recurso está ocupado o no.

El evento de parar el sistema es el que lleva al estado de reposo o *Stand-by*. El fin de este es eliminar cualquier evento de salida, añadiendo el paquete correspondiente a la lista de espera o cola, liberando así al recurso; además, debe generar un evento para restablecer el sistema. Este segundo evento, como su propio nombre indica, restaura el procesado de paquetes. El primer paso es calcular el nuevo *split* a usar, para ello se sirve de las probabilidades de los mismos. A continuación, si en el evento anterior existía un cliente en el servicio –su proceso se "congeló"– , se replica de nuevo el evento de salida en el *split* correspondiente; si por el contrario hay elementos en cola pero no llegaron a entrar en el servicio, se realiza uno nuevo para el siguiente cliente en la cola. Para terminar sólo falta generar el próximo evento en el que el controlador volverá al modo *Stand-by*.

Pseudocódigo

Tras esta primera explicación se pasa al pseudocódigo, expuesto en el Algoritmo 3, donde se observan los cambios respecto al simulador del modelo clásico. Ello comienza en la línea 8, ya que si el sistema está en *Stand-by* o el recurso está ocupado, todas las llegadas tendrán que enviarse a la cola. Lo más importante sin duda empieza con los nuevos eventos protagonistas que controlan el movimiento de *splits*, entre las líneas 22 y 30.

Lo que ocurre en las líneas 24 y 25, donde se envía al cliente del recurso a la cola, se explicará en detalle en la siguiente sección, ya que para este caso es trivial al no tener una limitación en el *buffer* de espera.

Algorithm 3 Modelo Controlador sin pérdidas

Require: λ, s, $\mu_1, \mu_2, ..., \mu_s$, $\alpha_1, \alpha_2, ..., \alpha_s$, γ

1: **procedure** SIMULATORCONTROLLER
2: ProcessArrivalEvent:
3: setPktTime (SYSTEM-IN, pkt, time)
4: **if** $(numPkts < numEntries)$ **then**
5: generateEvent(ARRIVAL, time, nextPkt)
6: $numPkts{+}{+}$
7: **end if**
8: **if** $(resourceBusy = true$ **OR** $controllerActive = false)$ **then**
9: queuePush(pkt)
10: **else**
11: generateEvent(DEPARTURE, time, pkt)
12: setPktTime (SERVER-IN, pkt, time)
13: **end if**
14: ProcessDepartureEvent:
15: setPktTime (SYSTEM-OUT, pkt, time)
16: **if** $(queue > 0)$ **then**
17: generateEvent(DEPARTURE, time, nextPkt)
18: setPktTime (SERVER-IN, nextPkt, time)
19: **else**
20: $resourceBusy = false$
21: **end if**
22: ProcessStopEvent:
23: $controllerActive = false$
24: $resourceHasPkt = false$
25: **if** $(resourceBusy = true)$ **then**
26: $resourceHasPkt = true$
27: **end if**
28: removeEvent(DEPARTURE)
29: generateEvent(RESTART, time)
30: ProcessRestartEvent:
31: CalculateNewSplit()
32: $controllerActive = true$
33: **if** $(resourceHasPkt = true)$ **then**
34: generateEvent(DEPARTURE, time, pkt)
35: $resourceBusy = true$
36: **else if** $(queue > 0)$ **then**
37: generateEvent(DEPARTURE, time, nextPkt)
38: setPktTime (SERVER-IN, nextPkt, time)
39: **end if**
40: generateEvent(STOP, time)
41: **end procedure**

3.2.2. Con *Buffer* de Espera Finito

Se trata ahora el segundo ejemplo, tras haber detallado el caso sin pérdidas. En este escenario y como se adelantó en la Sección 3.1, se usará un sistema con *buffer* finito. El modo de operación del simulador vuelve a ser muy parecido al anterior, teniendo en cuenta el máximo número de paquetes que puedan estar esperando a ser cursados. La diferencia por ende, está en los eventos de llegada y, como se explicó en el M/M/1 truncado, funciona del siguiente modo: si el recurso está ocupado y los clientes en cola igualan al *buffer*, se descartará el nuevo paquete –incrementando así la probabilidad de pérdida.

Es importante comentar además lo que ocurre en los otros dos eventos específicos de este modelo, el de parado y reanudado del sistema, aunque no exista cambio en el código respecto al caso previo. Como se ha visto en la sección anterior, en el evento de parado se elimina el posible evento de salida y se envía su correspondiente paquete a la cola. Esto podría ocasionar un desborde si el *buffer* coincide con el número de paquetes en espera, pero lo que realmente ocurre en el simulador lo impide. Cada vez que hay un recurso en el servidor al ejecutarse un evento de parada, hace que se genere una copia del paquete en cuestión y se almacene, sin entrar a la cola. Es decir, se queda en el servidor pero sin cursarse (por ejemplo, en una caché), en un estado de *Stand-by*. Al realizar el evento para restablecer el sistema, el procesado de este paquete vuelve a retomarse.

Pseudocódigo

Por último, se presenta en el Algoritmo 4 el simulador para el caso controlador con pérdidas.

Se destacan las líneas 28-29 y 36-37, de los eventos de parada y reanudado respectivamente. En las primeras se genera la copia que permite mantener el paquete –que estaba en el recurso– en reposo, sin trasladarlo de manera que pudiera eliminar algún otro paquete ya en cola; en las segundas, se reanuda el procesado de este paquete y se cursa, creando para ello su evento correspondiente de salida.

Algorithm 4 Modelo Controlador con pérdidas

Require: λ, s, $buffer$, $\mu_1, \mu_2, ..., \mu_s$, $\alpha_1, \alpha_2, ..., \alpha_s$, γ

```
1: procedure SimulatorBufferController
2: ProcessArrivalEvent:
3:     setPktTime (SYSTEM-IN, pkt, time)
4:     if (numPkts < numEntries) then
5:         generateEvent(ARRIVAL, time, nextPkt)
6:         numPkts++
7:     end if
8:     if (resourceBusy = true OR controllerActive = false) then
9:         if (queue = buffer) then
10:            lostPkt++
11:            break
12:        end if
13:        queuePush(pkt)
14:    else
15:        generateEvent(DEPARTURE, time, pkt)
16:        setPktTime (SERVER-IN, pkt, time)
17:    end if
18: ProcessDepartureEvent:
19:    setPktTime (SYSTEM-OUT, pkt, time)
20:    if (queue > 0) then
21:        generateEvent(DEPARTURE, time, nextPkt)
22:        setPktTime (SERVER-IN, pkt, time)
23:    else
24:        resourceBusy = false
25:    end if
26: ProcessStopEvent:
27:    controllerActive = false
28:    resourceHasPkt = false
29:    if (resourceBusy = true) then
30:        resourceHasPkt = true
31:    end if
32:    removeEvent(DEPARTURE)
33:    generateEvent(RESTART, time)
34: ProcessRestartEvent:
35:    CalculateNewSplit()
36:    controllerActive = true
37:    if (resourceHasPkt = true) then
38:        generateEvent(DEPARTURE, time, pkt)
39:        resourceBusy = true
40:    else if (queue > 0) then
41:        generateEvent(DEPARTURE, time, nextPkt)
42:        setPktTime (SERVER-IN, nextPkt, time)
43:    end if
44:    generateEvent(STOP, time)
45: end procedure
```

Capítulo 4

Pruebas y resultados

En este capítulo se aborda la comprobación del correcto funcionamiento del simulador. Como se ha explicado, este es el punto más importante del trabajo. Para ello se vale de los modelos clásicos, de las fórmulas matemáticas y de sus parámetros de funcionamiento, vistos en la Sección 2.3. El simulador se encarga de validar dichos modelos, generando resultados para su comparación, como pueden ser la ocupación del sistema o los tiempos de procesado. Cuando se pueda asegurar que el funcionamiento es el adecuado, se analizará controlador vRAN utilizando el simulador. En este caso, se expondrá una batería de pruebas con escenarios más específicos, ya que se necesita hacer una validación extensa, comparando estos resultados con los obtenidos en el modelo teórico.

4.1. Modelos Clásicos

Esta sección se centrará en concluir si el simulador es válido. Para poder hacerlo, se simularán escenarios que concuerden con los modelos clásicos anteriormente descritos, como son el M/M/1 (y su variación con *buffer* finito) y el M/G/1.

Para el modelo M/M/1 y M/G/1 el simulador no cambia, algo que se ha mostrado en la Sección 3.1.1. La diferencia entre ambos se encuentra en el cálculo de los resultados de forma teórica, ya que su funcionamiento práctico se rige por las mismas características –se recuerda que esto ocurre porque el primero es en realidad una particularización del segundo. Por estos motivos, se prefiere dividir esta comprobación en ambos modelos, cada uno con los cálculos matemáticos que correspondientes. Además, para asegurar que el simulador funciona correctamente cuando hay pérdidas de paquetes, se generarán una serie de resultados con el M/M/1 truncado (con *buffer* finito).

Tabla 4.1: Escenarios del simulador en modelos clásicos.

Variable	Rango
λ	$\{0.1 : 2.5\}$
μ	15
Buffer	[2, 4, ∞]

Los escenarios a tratar se muestran en la Tabla 4.1. Sólo se necesita tener en cuenta tres variables: tasa de llegadas (λ), la tasa de servicio (μ^{-1}) y el tamaño del *buffer*. Obviamente el último parámetro solo será utilizado en la configuración para el modelo M/M/1 truncado. La diferencia entre los escenarios del M/M/1 y el M/G/1 se halla en la distribución del tiempo de servicio, la cual sigue una variable aleatoria exponencial en el primer modelo mientras que para el segundo, se opta por usar un valor constante y una distribución uniforme.

4.1.1. M/M/1

Se empieza con el modelo M/M/1. En este caso, para poder comprobar si el simulador funciona correctamente, se generarán resultados de tres propiedades básicas, como se puede ver en la Figura 4.1.

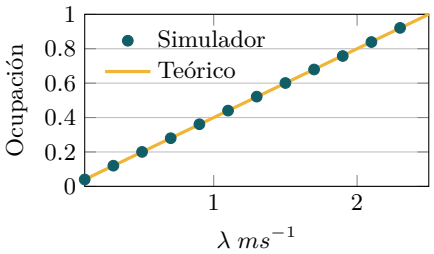

Figura 4.1.a: Ocupación M/M/1.

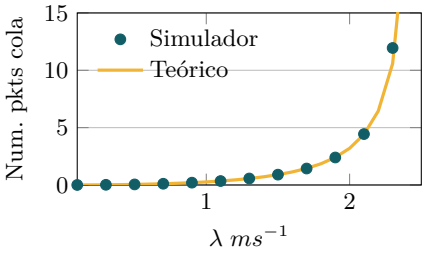

Figura 4.1.b: Número medio de paquetes en cola M/M/1.

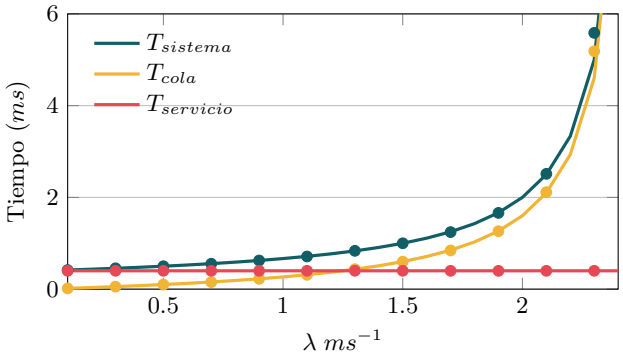

Figura 4.1.c: Tiempos medio M/M/1.

Figura 4.1: Resultados M/M/1. El simulador se muestra con marcadores, mientras que los resultados teóricos están en línea continua.

En la primera, la Figura 4.1.a, se comparan los valores teóricos frente a los del simulador de la ocupación del sistema ($\rho = \lambda/\mu$) respecto a la tasa de llegadas. A mayor tasa de llegadas, mayor es la ocupación. Como es una probabilidad, su valor máximo es 1 y esto ocurre cuando λ y μ coinciden. Si se supera, el sistema pasa a ser inestable y no se puede analizar con precisión; es por este motivo por lo que todos los resultados se calculan hasta ese valor. El simulador refleja a la perfección los resultados obtenidos con la fórmula matemática.

Posteriormente, se analiza la tasa de llegadas frente el número medio de paquetes existentes en la cola, en la Figura 4.1.b. En ambos casos, los resultados crecen exponencialmente como se podía suponer, ya que cuanto mayor es la ocupación del recurso, más paquetes deben estar esperando a ser cursados. Otro punto a tener en cuenta es que a mayor inestabilidad existente en el sistema, se podría haber esperado una mayor variación entre el simulador y los valores teóricos; sin embargo, la comparación vuelve a ser casi idéntica.

Por último, se estudia en la Figura 4.1.c los resultados obtenidos respecto a los tiempos medios de cada paquete en el sistema, siendo éste la suma de los otros dos: el tiempo medio que pasa un paquete en cola y el que transcurre cuando se cursa en el servicio o recurso. Se puede observar un incremento de manera exponencial también en este caso, causado por el crecimiento del tiempo medio en cola o tiempo de espera –el tiempo de servicio no varía con λ. Como en el caso anterior, esta tendencia es lógica, ya que se existen más clientes en espera, aumentando entonces dicho tiempo. El simulador vuelve a generar resultados muy semejantes al modelo teórico.

Tras realizar estas pruebas, y conseguir resultados tan parecidos a los teóricos, se llega a la conclusión de que el simulador funciona correctamente por caracterizar el comportamiento de un sistema clásico como es el M/M/1.

4.1.2. M/G/1

Para este modelo se van a comparar sólo los tiempos medio en el sistema, de la misma forma que en la última subfigura de la sección anterior, ya que los otros resultados no son relevantes para esta configuración –no cambian respecto al M/M/1. Como se explicó al comienzo de este capítulo, se van a analizar dos escenarios, donde solo se varía la distribución del tiempo de servicio: en el primero se usa una variable constante y en el segundo, una uniforme. Todo ello queda reflejado en la Figura 4.2.

Tanto la Figura 4.2.a como la Figura 4.2.b muestran un crecimiento exponencial y son similares entre sí –como era de esperar parecidos a los observados en la Figura 4.1.c. Se vuelve a obtener un tiempo de servicio constante, siendo el tiempo medio en espera el que ocasiona la subida. Esto es así, una vez más, por el aumento de paquetes en la cola y dicho crecimiento debido a una ocupación del recurso mayor. Asimismo, la comparación entre el simulador y los resultados de las fórmulas matemáticas del modelo teórico vuelven a ser prácticamente iguales –exceptuando cuando el sistema es inestable, donde los valores difieren levemente.

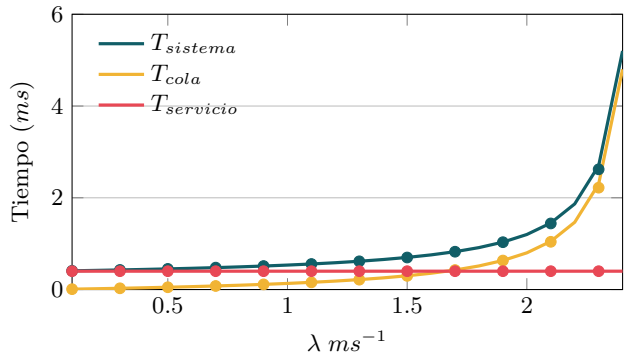

Figura 4.2.a: Distribución del tiempo de servicio con una variable constante.

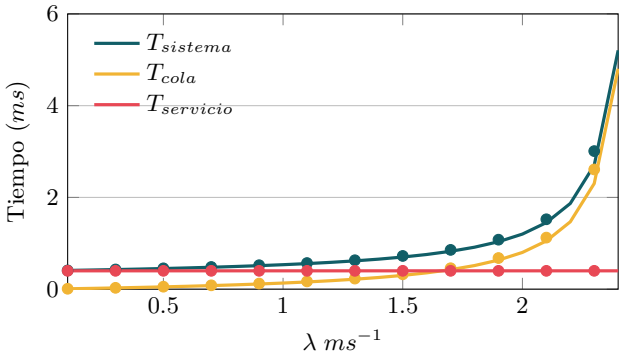

Figura 4.2.b: Distribución del tiempo de servicio con una variable uniforme.

Figura 4.2: Tiempo medio en el sistema M/G/1 con diferentes *v.a.* El simulador se muestra con marcadores, mientras que los resultados teóricos están en línea continua.

Habiendo comprobado el correcto comportamiento del simulador con ambos modelos clásicos, en configuraciones donde se estudia un sistema sin pérdidas, queda validado su funcionamiento en sistemas con *buffer* infinito.

4.1.3. M/M/1 truncado: con *Buffer* de Espera Finito

Por último, solo queda analizar cómo funciona el simulador en un sistema con pérdidas. Para ello y, como se ha explicado en el Capítulo 3, se utilizará un escenario con *buffer* finito. Como se está ante un sistema que no sigue un modelo clásico, sino que presenta ciertas modificaciones, la forma de verificarlo reside en encontrar las ecuaciones válidas de la probabilidad de pérdida de paquetes en el sistema. Conforme a lo anterior, se contemplan dos ejemplos, donde el primero trabaja con un *buffer* de 2 y el segundo, de 4.

Se presenta en la Figura 4.3 el primero de ellos, en el que el máximo número de paquetes simultáneos en cola puede ser 2. Su diagrama tiene por tanto 4 estados, siendo el primero cuando el sistema está vacío y el último, lleno –un paquete en el recurso y dos en cola.

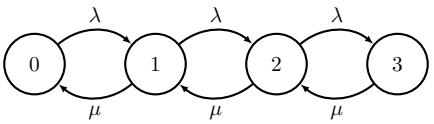

Figura 4.3: Diagrama de estados M/M/1 Truncado: *Buffer* = 2

La probabilidad de pérdida coincide con la del estado en el que no pueda recibir más paquetes, en este caso, el 3. Para poder obtener dicho resultado se debe resolver el siguiente sistema de ecuaciones:

$$\left.\begin{array}{l} P_2 \cdot \lambda = P_3 \cdot \mu \rightarrow P_3 = \rho \cdot P_2 = \rho^3 \cdot P_0 \\ P_1 \cdot \lambda = P_2 \cdot \mu \rightarrow P_2 = \rho \cdot P_1 = \rho^2 \cdot P_0 \\ P_0 \cdot \lambda = P_1 \cdot \mu \rightarrow P_1 = \rho \cdot P_0 \\ \sum_{i=0}^{3} P_i = 1 \end{array}\right\} \begin{array}{l} P_0 \cdot (1 + \rho + \rho^2 + \rho^3) = 1 \\ P_0 = \dfrac{1}{1 + \rho + \rho^2 + \rho^3} \end{array}$$

$$Prob.Pérdida = P_3 = \rho^3 \cdot P_0 = \frac{\rho^3}{1 + \rho + \rho^2 + \rho^3}$$

Una vez obtenida la fórmula del primer ejemplo, se pretende hacer lo mismo con el segundo. En este nuevo sistema, el cual se representa en la Figura 4.4, el *buffer* se incrementa en 2, llegando al total de 4 paquetes simultáneos en cola como máximo.

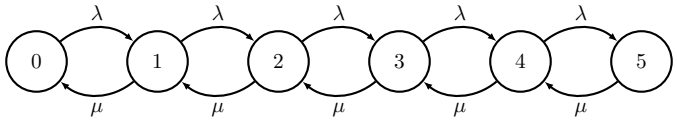

Figura 4.4: Diagrama de estados M/M/1 Truncado: *Buffer* = 4

Para el cálculo de la probabilidad de pérdida hay que fijarse en el último estado, el 5, al igual que ocurría en el ejemplo anterior. De nuevo, se analiza el diagrama para obtener las ecuaciones necesarias hasta la deseada:

$$\left.\begin{array}{l} P_4 \cdot \lambda = P_5 \cdot \mu \rightarrow P_5 = \rho \cdot P_4 = \rho^5 \cdot P_0 \\ P_3 \cdot \lambda = P_4 \cdot \mu \rightarrow P_4 = \rho \cdot P_3 = \rho^4 \cdot P_0 \\ P_2 \cdot \lambda = P_3 \cdot \mu \rightarrow P_3 = \rho \cdot P_2 = \rho^3 \cdot P_0 \\ P_1 \cdot \lambda = P_2 \cdot \mu \rightarrow P_2 = \rho \cdot P_1 = \rho^2 \cdot P_0 \\ P_0 \cdot \lambda = P_1 \cdot \mu \rightarrow P_1 = \rho \cdot P_0 \\ \sum_{i=0}^{3} P_i = 1 \end{array}\right\} \quad P_0 = \frac{1}{1 + \rho + \rho^2 + \rho^3 + \rho^4 + \rho^5}$$

$$Prob.Pérdida = P_5 = \rho^5 \cdot P_0 = \frac{\rho^5}{1 + \rho + \rho^2 + \rho^3 + \rho^4 + \rho^5}$$

Se comparan en la Figura 4.5 los resultados generados tanto por el simulador como los teóricos, los segundos calculados gracias a las ecuaciones obtenidas con anterioridad. Como era de suponer, la probabilidad de pérdida crece con la ocupación del sistema. Pero también hay que tener en cuenta el valor del *buffer*; cuanto menor es este número, menor son clientes que pueden estar esperando a ser procesados y, por tanto, mayor es la pérdida de paquetes –esto es visible al comparar las Figuras 4.5.a y 4.5.b.

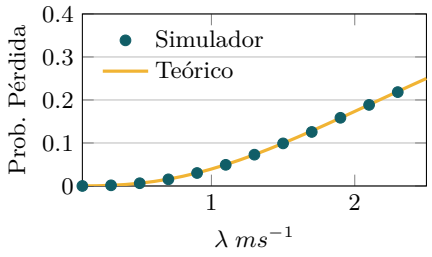

Figura 4.5.a: Ejemplo 1. *Buffer* = 2

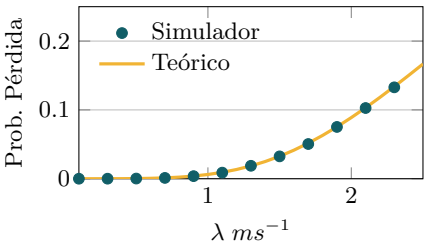

Figura 4.5.b: Ejemplo 2. *Buffer* = 4

Figura 4.5: Probabilidad de pérdida M/M/1 truncado. El simulador se muestra con marcadores, mientras que los resultados teóricos están en línea continua.

Como la similitud entre los resultados del simulador y de los cálculos teóricos es excelente, se asegura el buen funcionamiento del programa con sistemas con pérdida de paquetes. Tras comprobar todos los escenarios de los modelos clásicos, se utiliza a continuación el simulador para analizar el comportamiento del controlador vRAN con *buffer* tanto finito como infinito.

4.2. Modelo Controlador

El objetivo de esta sección es validar el modelo controlador, cuyo funcionamiento fue explicado en la Sección 3.2. En este caso, se realizará una batería de pruebas más extensa, cuyas características se pueden ver en la Tabla 4.2. Existen cuatro escenarios, diferenciándose en el número de *splits* (2 o 4) y en las probabilidades de uso (α_i) de cada uno de ellos. Para ampliar el rango de resultados, se usarán tasas de llegadas[1] (λ) diferentes. Igualmente, los escenarios se adaptarán al *buffer* de cada apartado, ya sea infinito para el sistema sin pérdidas, como finito cuando las hay. Otro punto a tener en cuenta es la tasa de servicio, ya que en los escenarios pares este valor aumenta a la par que su probabilidad, generando sistemas más rápidos, mientras que en los impares pasa lo contrario. Los últimos valores quedan constantes para todos los escenarios –se recuerda que el primero (γ) corresponde al tiempo medio ante cambios de *split*, mientras que el segundo (ξ) es el inverso del tiempo transcurrido en el modo de reposo o *Stand-by*.

[1]Donde se compara el valor de λ con otras características como son el retardo o la probabilidad de pérdida, se usarán los valores referentes al rango $\{0.1 : 2\}$. En cambio, donde no se especifique, se usará el valor correspondiente reflejado en la Tabla 4.2.

Tabla 4.2: Escenarios del simulador para analizar el controlador vRAN.

Escenario	#*Splits*	Buffer	λ	α	μ	γ	ξ
A	2	[4, 8, 16, 32, ∞]	1	[0.75, 0.25]	[1, 4]	0.1	1
B	2	[4, 8, 16, 32, ∞]	1.5	[0.25, 0.75]	[1, 4]	0.1	1
C	4	[4, 8, 16, 32, ∞]	1	[0.4, 0.3, 0.2, 0.1]	[1, 1.5, 3, 5]	0.1	1
D	4	[4, 8, 16, 32, ∞]	1.5	[0.1, 0.2, 0.3, 0.4]	[1, 1.5, 3, 5]	0.1	1

4.2.1. Con *Buffer* de Espera Infinito

El primer punto será validar el modelo para un sistema con espera pura, es decir, los escenarios analizados asumen un *buffer* infinito. Para empezar, se estudiarán los resultados obtenidos tanto por el modelo como del simulador sobre la probabilidad de tener n paquetes en el controlador, tal y como muestra la Figura 4.6. En otras palabras, averiguar la probabilidad de estar en un estado u otro, siendo el primero cuando el sistema está vacío.

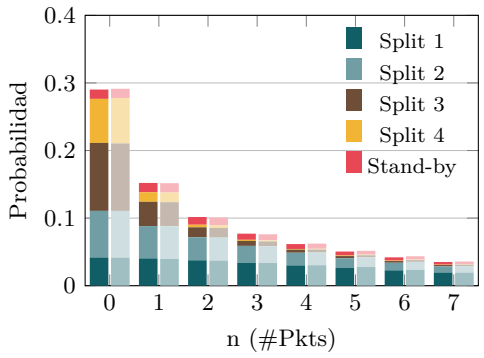

Figura 4.6.a: Escenario C

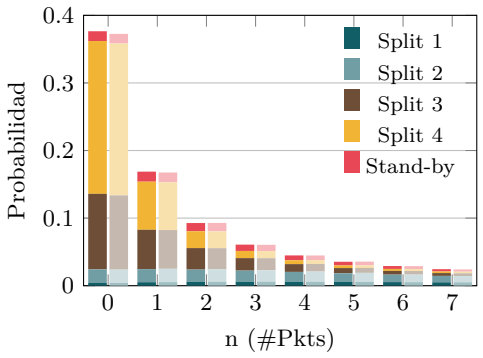

Figura 4.6.b: Escenario D

Figura 4.6: Probabilidad de tener n paquetes en el controlador con *buffer* infinito. El modelo se muestra con barras sólidas, mientras que el simulador está sombreado.

Lo primero, y más notable, es la similitud entre resultados del modelo y el simulador, los cuales prácticamente no varían entre sí. Además, se recuerda que en la Figura 4.6.a el sistema es más lento ya que las probabilidades de los *splits* con tasas de tiempo de servicio más elevadas son menores, cuya consecuencia es visible al tener probabilidades más elevadas en estados con mayor número de paquetes en el controlador. Al contrario ocurre en la Figura 4.6.b, donde la probabilidad de tener el sistema vacío aumenta notablemente.

El siguiente elemento a analizar es el tiempo medio del sistema, representado en la Figura 4.7. Al igual que ocurría en los modelos clásicos, los valores crecen exponencialmente, debido a la mayor ocupación del recurso y, también, al incremento de paquetes en cola. Es importante tener en cuenta además la diferencia entre los escenarios, donde se observa que el tiempo medio en el controlador aumenta considerablemente con λ – siendo más relevante para los escenarios A y C, cuyo rango que garantiza un régimen estable es más corto. Una vez más se consiguen resultados que no varían significativamente entre el modelo y el simulador. Destaca además que el comportamiento con el tiempo de servicio es constante.

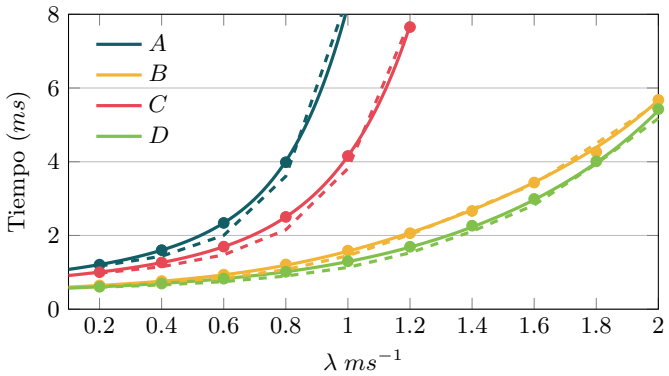

Figura 4.7: Tiempo medio en el sistema del modelo controlador con *buffer* infinito. El modelo se muestra con línea continua. El simulador se representa tanto con línea discontinua como con marcadores, utilizando tasas de tiempo de servicio constantes y exponenciales negativas, respectivamente.

Se ha podido comprobar por una parte que el simulador se comporta correctamente con modelos más complejos y, por otra, la propia validación del modelo del controlador vRAN en sistemas sin pérdidas.

4.2.2. Con *Buffer* de Espera Finito

Para poder corroborar en profundidad el funcionamiento del modelo teórico, se amplían los escenarios de manera que introduzcan pérdidas. Se vuelve a calcular, en primer lugar, la probabilidad de estar en cada estado o, lo que es lo mismo, de tener n paquetes en el controlador. Como se refleja en la Figura 4.8 existe un último estado donde el sistema está lleno –ocho paquetes en espera y uno cursado–; en comparación al sistema sin pérdidas, en el cual existían infinitos estados.

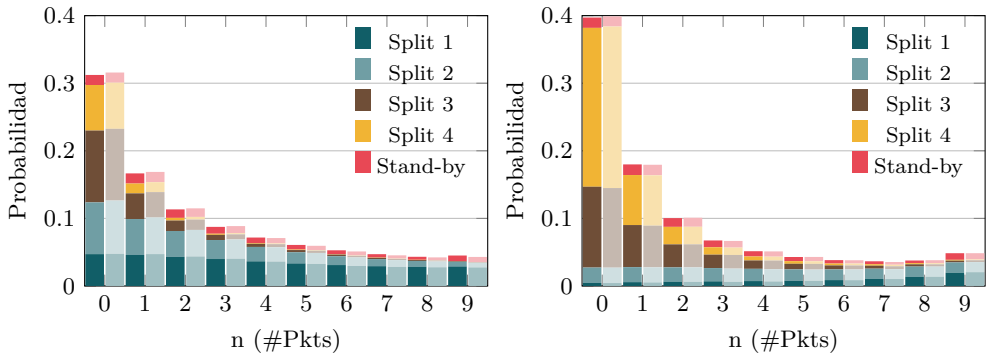

Figura 4.8.a: Escenario C. *Buffer* = 8.

Figura 4.8.b: Escenario D. *Buffer* = 8.

Figura 4.8: Probabilidad de tener n paquetes en el controlador con *buffer* finito. El modelo se muestra con barras sólidas, mientras que el simulador está sombreado.

Como ocurría en la configuración de la Figura 4.8.b, el controlador opera sobre todo en los primeros estados, donde hay menos paquetes, al contrario de lo que se ve en la Figura 4.8.a. Esto es debido, una vez más, a la velocidad de los sistemas en procesar recursos, es decir, de utilizar tasas de servicio más elevadas. Aparte de este hecho, se puede visualizar cómo los resultados entre el modelo teórico y el simulador son prácticamente idénticos.

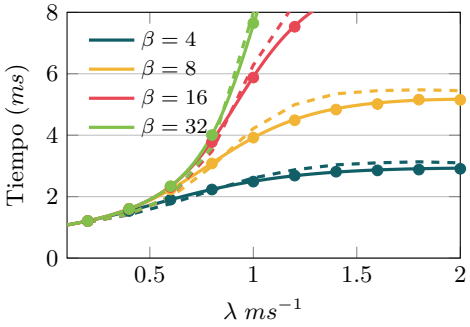

Figura 4.9.a: Escenario A

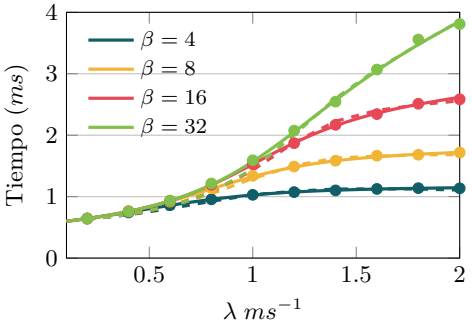

Figura 4.9.b: Escenario B

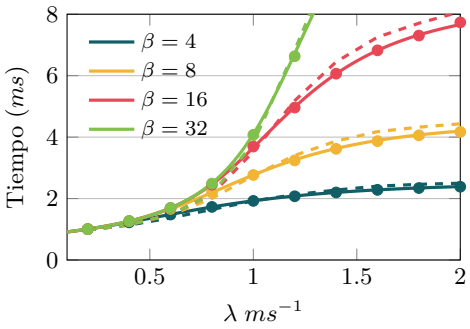

Figura 4.9.c: Escenario C

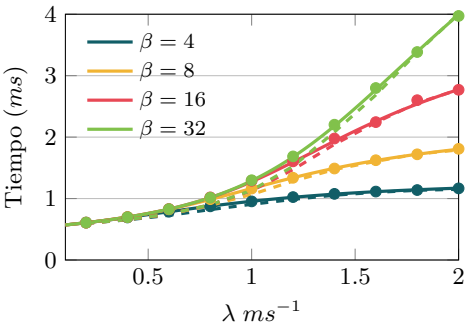

Figura 4.9.d: Escenario D

Figura 4.9: Tiempo medio en el sistema del modelo controlador con *buffer* finito. El modelo se muestra con línea continua. El simulador se representa tanto con línea discontinua como con marcadores, utilizando tasas de tiempo de servicio constantes y exponenciales negativas, respectivamente.

Se estudia a continuación la comparativa entre el modelo y el simulador generando los resultados del tiempo medio en el sistema, que se muestran en la Figura 4.9. Para empezar, se consiguen resultados idénticos de nuevo. Se vuelve a poner de manifiesto una diferencia muy escasa con la tasa de servicio constante –se puede notar cierta variación cuanto más lento es el sistema, como ocurre en la Figura 4.9.a y la Figura 4.9.c. Si se observan las figuras con atención, se ve que el tiempo tiende a estabilizarse cuanto mayor es λ y que, además, sufre un aumento considerable con la capacidad del *buffer*. Estos resultados tienen sentido ya que, cuanto menor es el número de paquetes que pueden estar esperando a ser cursados, más pérdida existe en el sistema, lo que lleva a menos clientes procesados. Obviamente, al existir menos paquetes que cursar, ese número se estabiliza y con él, el tiempo medio en el sistema –este hecho es más visible en la Figura 4.9.b y Figura 4.9.d. Al contrario, con un *buffer* elevado, se asemeja a lo ocurrido en un sistema sin pérdidas.

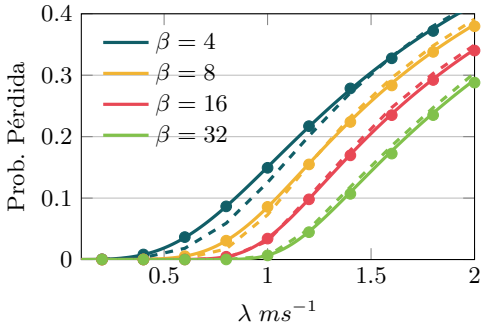

Figura 4.10.a: Escenario A

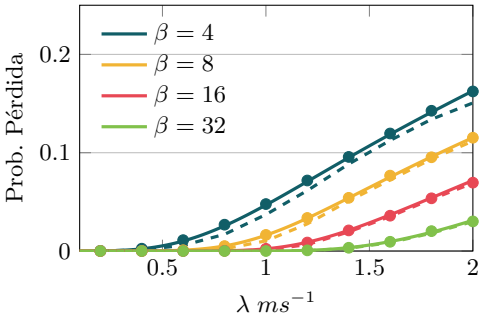

Figura 4.10.b: Escenario B

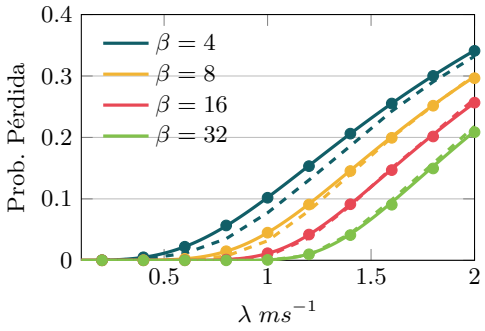

Figura 4.10.c: Escenario C

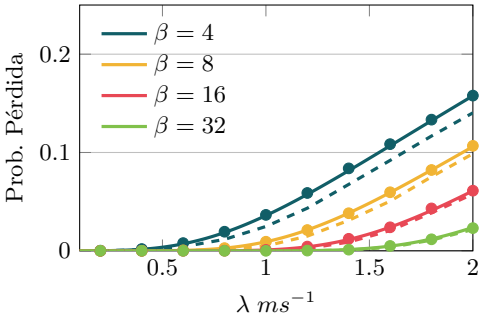

Figura 4.10.d: Escenario D

Figura 4.10: Probabilidad de pérdida en el sistema modelo controlador. El modelo se muestra con línea continua. El simulador se representa tanto con línea discontinua como con marcadores, utilizando tasas de tiempo de servicio constantes y exponenciales negativas, respectivamente.

Otro factor con bastante importancia es la probabilidad de pérdida, la cual se muestra, para los distintos escenarios, en la Figura 4.10. Se vuelve a poner de manifiesto que los resultados de forma teórica y práctica coinciden a la perfección, existiendo una ligera diferencia cuando se utiliza una tasa de servicio constante –más notable cuanto menor es el *buffer*. Como era de esperar, cuanto mayor es la tasa de llegadas λ, mayor es la probabilidad de pérdida. Además, se incrementa con el menor número de paquetes en espera, ya que como se explicó con anterioridad, esto implica menos paquetes en el sistema. Asimismo, la velocidad del recurso al cursar un paquete también juega un papel importante, como se observa si se comparan la Figura 4.10.a con la Figura 4.10.b o la Figura 4.10.c con la Figura 4.10.d.

Como se ha demostrado, a mayor tamaño de *buffer*, menor probabilidad de perder paquetes. Pero anteriormente se llegó a la conclusión de que a mayor número de paquetes permitidos en espera, mayor es el tiempo medio transcurrido en el sistema. Por lo tanto, es conveniente estudiar la relación entre la probabilidad de pérdida y el tiempo medio, mostrado en la Figura 4.11. En ella, se analizan tres tasas de llegadas ($\lambda = [1, 1.5, 2]$) y se aumenta el tamaño de *buffer*. Para empezar, se puede ver una diferencia clara entre los escenarios con configuraciones más lentas, la Figura 4.11.a y la Figura 4.11.c, y las más rápidas, la Figura 4.11.b

y la Figura 4.11.d. En el primer grupo, al operar con λ mayores y mayor tamaño del *buffer*, se produce una leve disminución de la probabilidad de pérdida (más notable en el escenario con mayor número de *splits*) pero con un aumento muy importante en el tiempo transcurrido en el sistema. Sin embargo, en el segundo grupo, al operar de la misma manera se consiguen resultados totalmente diferentes; el aumento del retardo en el sistema casi no es relevante, mientras que hay una fuerte disminución de la probabilidad de pérdida. Una vez más se consiguen resultados muy similares entre los teóricos y los del simulador, teniendo en cuenta una pequeña variación en las tasas de llegada y tamaño de *buffer* elevados –lo cual significa que el rendimiento es menos predecible.

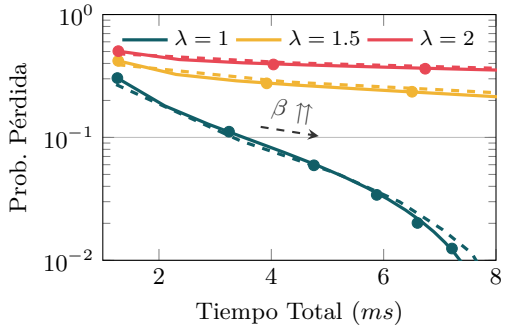

Figura 4.11.a: Escenario A

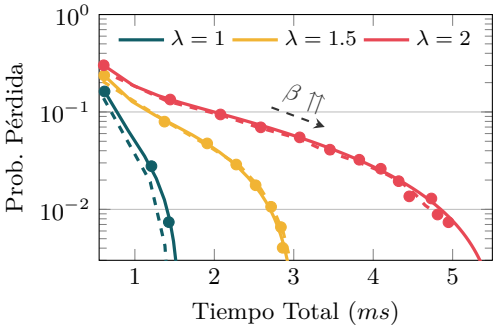

Figura 4.11.b: Escenario B

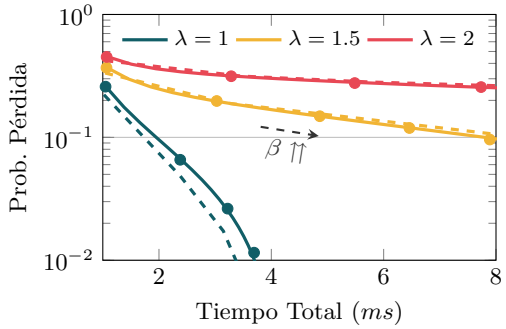

Figura 4.11.c: Escenario C

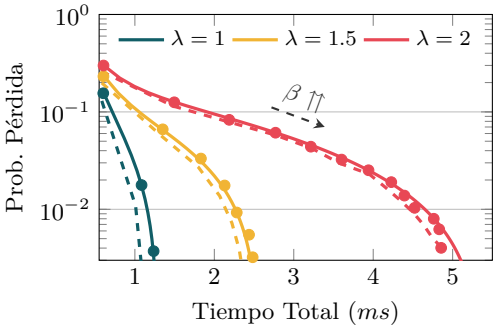

Figura 4.11.d: Escenario D

Figura 4.11: Probabilidad de pérdida en comparación al tiempo total en el sistema, se incrementa el tamaño del *buffer*. El modelo se muestra con línea continua. El simulador se representa tanto con línea discontinua como con marcadores, utilizando tasas de tiempo de servicio constantes y exponenciales negativas, respectivamente.

Como todos los valores que se han analizado entre el modelo teórico y el simulador coinciden casi de manera perfecta, se valida por tanto el modelo del controlador vRAN (tanto en sistemas con o sin pérdida de paquetes). Sin embargo, el modelo teórico solo ofrece resultados promedios, por lo que se explota el simulador para evaluar algunos indicadores de rendimiento como la variabilidad del tiempo que pasan los paquetes en el sistema o la de la probabilidad de pérdida, representados en la Figura 4.12 y la Figura 4.13, respectivamente. En ellas, cada *boxsplot* corresponde a un único experimento y muestran: (1) la mediana del tiempo, es decir, el percentil 0.5 (línea horizontal dentro de la caja); (2) los percen-

tiles 0.25 y 0.75 (límites de la caja inferior y superior, respectivamente); y (3) los percentiles 0.05 y 0.95 (líneas inferior y supererior, respectivamente, fuera de la propia caja). El marcador circular en el interior de la caja simboliza el valor medio, el cual ha sido estudiado en las figuras anteriores. Se analizan los resultados para los escenarios B y D, tanto con tiempos de servicio distribuidos con una exponencial negativa como con una constante.

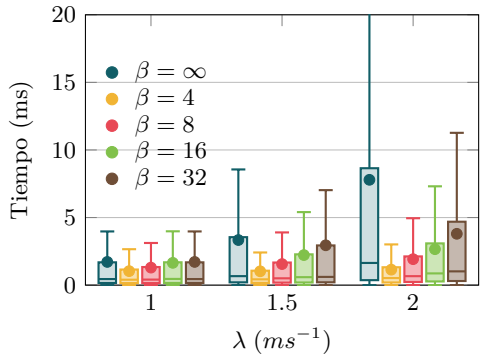

Figura 4.12.a: $T_{servicio}$ exponencial negativo. Escenario B.

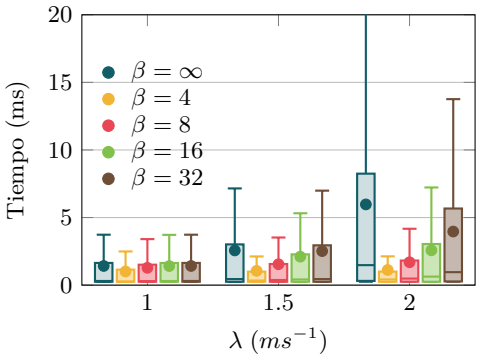

Figura 4.12.b: $T_{servicio}$ constante. Escenario B.

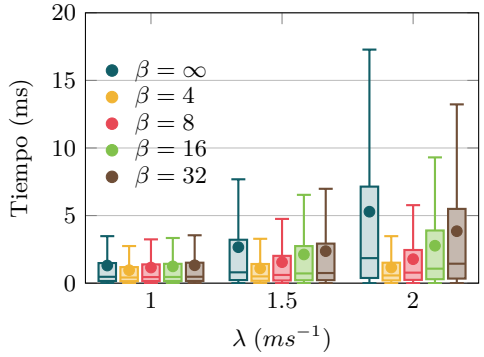

Figura 4.12.c: $T_{servicio}$ exponencial negativo. Escenario D.

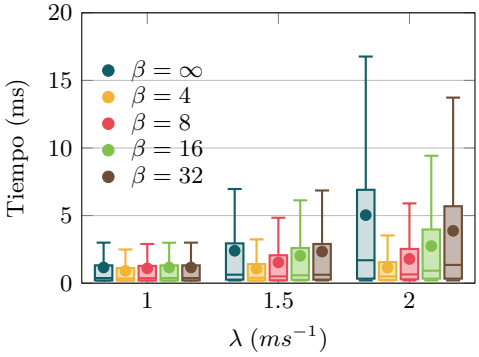

Figura 4.12.d: $T_{servicio}$ constante. Escenario D.

Figura 4.12: Boxplots sobre la variabilidad del tiempo en el sistema.

Se estudia primero el retardo del sistema, tanto con *buffer* finito como infinito. Como se puede ver en la Figura 4.12, existe una correlación directa entre el valor de λ y la variabilidad. Además, cuando el valor de λ es menor, la variabilidad casi no depende del tamaño del *buffer* elegido. En cambio, en el caso contrario al aumentar la tasa de llegadas, la variabilidad respecto al tiempo de permanencia en el controlador es considerablemente mayor y se hace más evidente cuanto mayor es el número de paquetes que puede haber en espera. En el sistema sin pérdidas (cuando el *buffer* es infinito) es donde se encuentra la mayor variabilidad, lo cual se puede observar en todos los escenarios. Los resultados no varían en exceso entre las diferentes configuraciones del tiempo de servicio. Sin embargo, sí existe una leve diferencia con un mayor o menor número de *splits* disponibles, donde en el primer caso se produce una menor variabilidad que en el segundo.

Y por último, se estudia en la Figura 4.13 la variabilidad de la probabilidad de pérdida, en este caso solo en el sistema con *buffer* finito, por motivos obvios. Se vuelve a encontrar una correlación directa entre la variabilidad, en este caso de la probabilidad de pérdida, y la tasa de llegadas. Sin embargo, al aumentar λ no se produce un aumento tan relevante en la variabilidad como sí ocurrió en la figura anterior. Otra peculiaridad respecto al análisis anteriormente realizado, es el tamaño del *buffer*, ya que para esta gráfica casi no tiene importancia. Como se puede observar en cualquier escenario, un aumento de los paquetes posibles en espera no implica una variabilidad más elevada. Asismismo ocurre al cambiar la distribución de la tasa de servicio o el número de *splits* en el sistema, ya que no hay correlación clara con la varianza de la probabilidad de pérdida.

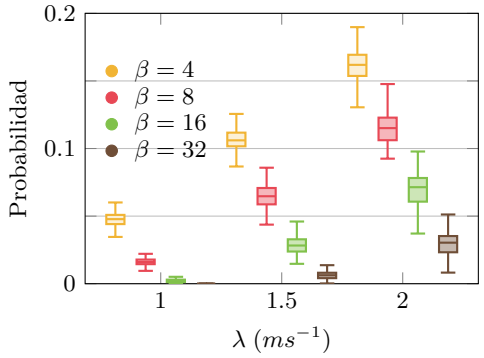

Figura 4.13.a: Prob. Pérdida exponencial negativa. Escenario B.

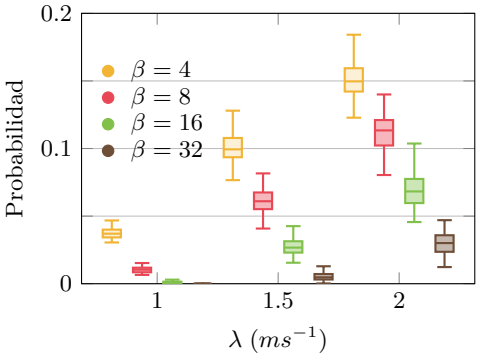

Figura 4.13.b: Prob. Pérdida constante. Escenario D.

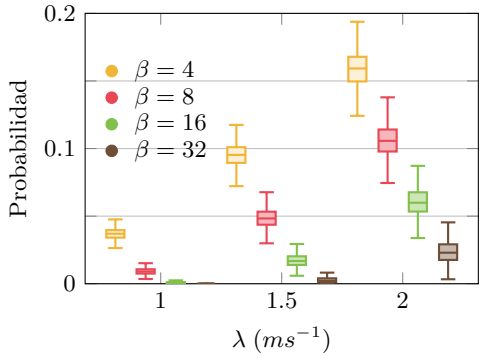

Figura 4.13.c: Prob. Pérdida exponencial negativa. Escenario B.

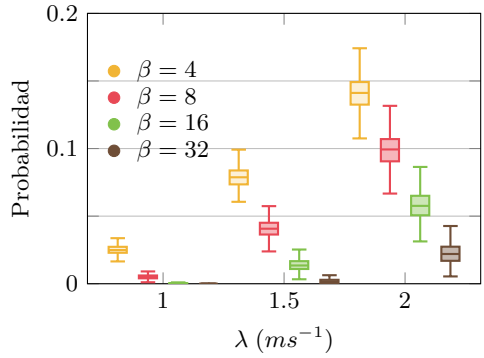

Figura 4.13.d: Prob. Pérdida constante. Escenario D.

Figura 4.13: Boxplots sobre la variabilidad de la probabilidad de pérdida en el sistema.

Capítulo 5

Conclusiones

Para la realización de este proyecto se han analizado diversos modelos de teletráfico, ya sean modelos clásicos, como el M/M/1 o el M/G/1 hasta modelos innovadores como lo es el de un controlador vRAN. El primer paso obligatorio entonces es entender cómo es su funcionamiento, siendo para el último caso necesario centrarse en la utilidad y el manejo de arquitecturas virtualizadas C-RAN y del propio *split* funcional flexible. Valiéndose de ambos elementos se consigue una situación ideal para cumplir con los exigentes requisitos que busca alcanzar el 5G: una arquitectura en la nube y centralizada, que ayuda a combatir la limitación de usuarios y de velocidad que se tenía en la generación anterior. El modelo propuesto se basa en esta postura, con la finalidad de analizar el rendimiento de los controladores vRAN 5G y así, abordar un dimensionado óptimo para cada servicio. Tras ello se ha examinado en profundidad cómo se implementa dicha idea, centrándonos en el manejo por parte del controlador de la variación entre los diferentes *splits* funcionales existentes en ese momento, sirviéndose para ello de la configuración *Stand-by*. Su novedad está en el estado de reposo del mismo, donde el controlador tiene la capacidad de poder calcular qué *split* es el siguiente. Mediante este trabajo se ha logrado apreciar los cambios en el rendimiento y en los tiempos de espera al variar las capacidades del controlador. También se ha obtenido una relación entre la probabilidad de pérdida de paquetes y el tiempo total de transferencia en un sistema con *buffer* limitado; cuanto menor se quiere que sea esa probabilidad (trabajando con tamaños de *buffer* elevados), mayor es el tiempo medio de permanencia.

Tras el estudio de todos los aspectos previos, se concluye con la obtención de los resultados que se buscaban desde un principio. Se consigue un simulador funcional por eventos que ayuda de manera práctica a evaluar la validez de modelos de dimensionado de tráfico –tanto clásicos como otros más avanzados. Se corrobora entonces el funcionamiento del modelo del controlador vRAN gracias a una amplia campaña de simulación donde se muestran unos resultados que coinciden casi a la perfección con los obtenidos de manera teórica. Además, se ha explotado el mismo, ampliando así el análisis, realizando cambios en los indicadores de rendimiento del controlador a usar.

5.1. Contribución

Este trabajo se enmarca dentro de la línea de investigación del Grupo de Ingeniería Telemática del Departamento de Ingeniería de Comunicaciones (DICOM) de la Universidad de Cantabria, en el cual se quiere analizar el rendimiento de la tecnología ligada al *split* funcional flexible en redes 5G debido a su creciente desarrollo en los últimos años. Mientras que unos trabajos se han centrado en las distintas divisiones posibles al realizar esta arquitectura y otros en intentar que esta división se haga de forma dinámica, en este caso se busca hallar la manera de capturar el comportamiento del controlador sobre una política en particular. Para poder validar el mismo se ha diseñado e implementado el simulador, cumpliendo con los requisitos de este nuevo modelo, y suponiendo dos escenarios que ya se han comentado: un modelo sin pérdidas y otro con ellas.

En ambos casos la diferencia de los resultados entre los valores teóricos y los del simulador es prácticamente nula. Es así como se ha podido validar el funcionamiento de este modelo, aunque no solo ha servido para ello. El modelo teórico permite únicamente conseguir resultados de valores medios, sin embargo, el simulador puede explotarse de forma que capture la variabilidad de los indicadores de rendimiento así como diferentes distribuciones de la tasa de servicio. Es por este motivo gracias al cual se han obtenido una serie de conclusiones: (1) a bajas tasas medias de llegadas (λ) la variabilidad del tiempo en el sistema es casi nula, mientras que cuanto más se aumenta λ, más varía el retardo –especialmente cuanto mayor es el *buffer*, sobre todo en el modelo sin pérdidas; (2) al realizar el mismo análisis de λ pero frente a la probabilidad de pérdida de paquetes, se observa esta variabilidad en menor medida –incluso en este caso, casi no se tiene en cuenta el tamaño del *buffer*; y (3) por último, lo anterior se cumple casi de la misma manera tanto en un régimen de tiempos de servicio constantes como exponenciales negativos, mientras que si se varía el número de *splits* en el sistema se genera una respuesta en la variabilidad –existe una correlación inversa: cuanto menor sea ese número, mayor es la variabilidad de ambos indicadores, siendo más notable en el primero.

Estas contribuciones han ayudado a llevar a cabo un par de publicaciones, actualmente en proceso de revisión, en el marco del proyecto Future Internet Enabled Resilient smart CitiEs (FIERCE), financiado por la Agencia Española de Investigación, las cuales se muestran en [30] y [31].

5.2. Trabajo futuro

A raíz de este trabajo aparecen una serie de líneas futuras. Uno de los puntos principales consiste en seguir explotando las funcionalidades del simulador. Se ha visto cómo se analizaban una serie de escenarios genéricos, pero se podrían estudiar otros casos de uso más específicos. De la misma forma, se pueden obtener más resultados relacionados con otros tiempos o probabilidades del sistema, dependiendo de cuáles son las necesidades del servicio a estudiar. Podría también ser interesante el analizar otros regímenes de entrada, ya que en este proyecto solo se ha tenido en cuenta según procesos de Poisson. Asimismo, se puede pro-

fundizar en la propia red y sus características, porque el simulador, al usarse para la validación del modelo controlador, se ha centrado en las capacidades del *split*.

El modelo del controlador vRAN es un elemento vital para este trabajo, y se puede explotar más, adentrándose en su gestión de los recursos computacionales. Una forma sería variar los tiempos en los que el controlador debe cambiar de *split*, y no trabajando con distribuciones exponenciales como en este trabajo. Otra posibilidad sería estudiar un procesado simultáneo de tramas o paquetes, mejorando así los tiempos de espera del sistema. También se puede enfocar al simulador en otro punto de la arquitectura, como son las RRH intermedias. En este caso, la utilidad va ligada al encaminado de las tramas y cómo poder gestionarlas, ya que son cruciales a la hora de optimizar la red. Otro punto a tener en cuenta es el análisis de otras arquitecturas, dejando de lado las redes vRAN Coordinadas como es la de este caso.

Al mismo tiempo, se desarrolla cómo gestionar las divisiones de los *splits* funcionales. En cambio, hay varios proyectos que analizan tanto la posibilidad de optimizar el desarrollo de separación de *splits* como en la búsqueda de técnicas de control dinámico, enfocado en *splits* flexibles. Adicionalmente, se considera la aplicación de diferentes políticas de *scheduling* –nuevos algoritmos para asignar recursos a las diferentes tareas– y técnicas de control dinámico de colas.

Bibliografía

[1] Salima Samaoui,Imen El Bouabidi, Mohammad S.Obaidat, Faouzi Zarai y Wahida Mansouri. "Wireless and mobile technologies and protocols and their performance evaluation". En: *Modeling and Simulation of Computer Networks and Systems* (Abril de 2015), págs 3-32. ISBN: 978-0-12-800887-4. DOI: 10.1016/C2013-0-19024-1.

[2] Luis Muñoz. "El Sistema GSM". En: *Redes Inalámbricas*. URL: https://www.tlmat.unican.es/siteadmin/submaterials/3193.pdf (visitado 19-04-2020).

[3] Luis Muñoz. "El Sistema UMTS". En: *Redes Inalámbricas*. URL: https://www.tlmat.unican.es/siteadmin/submaterials/3211.pdf (visitado 12-05-2020).

[4] Nan Jiang, Yu Jin, Ann Skudlark, Wen-Ling Hsu, Guy Jacobson, Siva Prakasam y Zhi-Li Zhang. "Isolating and analyzing fraud activities in a large cellular network via voice call graph analysis". En: *MobiSys'12 - Proceedings of the 10th International Conference on Mobile Systems, Applications and Services* (Junio 2012). DOI: 10.1145/2307636.2307660.

[5] María Eugenia González. "Capítulo 1. La Tecnología UMTS". En: *Sistemas de Localización en UMTS*. URL: http://bibing.us.es/proyectos/abreproy/11844/fichero/Cap%C3%ADtulo1.+La+Tecnolog%C3%ADa+UMTS.pdf+ (visitado 16-05-2020).

[6] Moe Rahnema. "UMTS System and Air Interface Architecture". En: *UMTS Network Planning, Optimization, and Inter-Operation with GSM* (Abril 2008), págs 7-38. ISBN: 978-0-470-82302-6. DOI: 10.1145/2307636.2307660.

[7] 3GPP. "IMT-2000 CDMA Direct Spread". En: *ITU-R Recommendation M.1457*. URL: ,https://www.3gpp.org/news-events/3gpp-wiki/2-uncategorised/1984-umts-detailed-description (visitado 17-05-2020).

[8] David Fajardo. "Capítulo 1. Estructura de la red UMTS". En: *Simulación de tramas de WCDMA*. URL: http://catarina.udlap.mx/u_dl_a/tales/documentos/lem/fajardo_p_d/capitulo1.pdf (visitado 17-05-2020).

[9] Shadi Nashwan. "SAK-AKA: A Secure Anonymity Key of Authentication and Key Agreement protocol for LTE network". En: *International Arab Journal of Information Technology Vol.14 No.5* (Septiembre 2017), págs 790-800. URL: https://www.researchgate.net/publication/318502441_SAK-AKA_A_Secure_Anonymity_Key_of_Authentication_and_Key_Agreement_protocol_for_LTE_network (visitado 18-05-2020).

[10] Sumit Kasera y Nishit Narang. "Fourth Generation (4G) Mobile Networks". En: *3G Networks: Architecture, Protocols and Procedures* (2004), págs 7-38. ISBN: 978-0-070-52799-7. DOI: 10.1145/2307636.2307660.

[11] Gonzalo López. "Capítulo 2: LTE o la Cuarta Generación (4G) de Comunicaciones Móviles". En: *Predistorsión Digital de un Sistema de Comunicaciones OFDM: Estudio mediante Simulación*. URL: http://bibing.us.es/proyectos/abreproy/11983/fichero/Cap%C3%ADtulo+2+-+LTE.pdf (visitado 18-05-2020).

[12] 3GPP. "LTE". En: *3GPP Release 8*. URL: https://www.3gpp.org/technologies/keywords-acronyms/98-lte (visitado 18-05-2020).

[13] Rupendra Mitra y Dharma Agrawal. "5G mobile technology: A survey". En: *ICT Express* (Enero 2016). DOI: 10.1016/j.icte.2016.01.003.

[14] Akhil Gupta and Rakesh Kumar. "A Survey of 5G Network: Architecture and Emerging Technologies". En: *IEE Access. Special Section on Recent Advances in Software Defined Networking for 5G Networks* (Agosto 2015). DOI: 10.1109/ACCESS.2015.2461602.

[15] Javier Madrazo. "Análisis y Estudio de Soluciones para Fronthaul Radio". En: *Repositorio Institucional (O2)*. URL: http://openaccess.uoc.edu/webapps/o2/bitstream/10609/72005/8/javimloriteTFM0118presentaci%C3%B3n.pdf (visitado 23-05-2020).

[16] Ramón Agüero. "Algoritmos y protocolos de nivel de red". En: *Diseño y Operación de Redes Telemáticas*. URL: https://www.tlmat.unican.es/siteadmin/submaterials/3070.pdf (visitado 23-05-2020).

[17] A. Checko, H. L. Christiansen, Y. Yan, L. Scolari, G. Kardaras, M. S. Berger y L. Dittmann. "Cloud ran for mobile networks—a technology overview". En: *IEEE Communications Surveys Tutorials, vol. 17, no. 1, Firstquarter* (2014), págs 405–426. DOI: 10.1109/COMST.2014.2355255.

[18] A. Hajisami, H. Viswanathan y D. Pompili. "*Cocktail Party in the Cloud*: Blind Source Separation for Co-operative Cellular Communication in Cloud RAN". En: *IEEE 11th International Conference on Mobile Ad Hoc and Sensor Systems* (2014), págs 37–45.

[19] G. O. Pérez, J. A. Hernández y D. Larrabeiti. "Fronthaul network modeling and dimensioning meeting ultra-low latency requirements for 5G". En: *IEEE/OSA Journal of Optical Communications and Networking, vol. 10, no. 6* (Junio 2018), págs 573–581. DOI: 10.1364/JOCN.10.000573.

[20] L. M. P. Larsen, A. Checko y H. L. Christiansen. "A Survey of the Functional Splits Proposed for 5G Mobile Crosshaul Networks". En: *IEEE Communications Surveys Tutorials 21.1* (Marzo 2019), págs 146-172. ISBN: 1553-877X. DOI: 10.1109/COMST.2018.2868805.

[21] D. Wübben, P. Rost, J. Barlelt, M. Lalam, V. Savin, M. Gorgoglione, A. Dekorsy y G. Fettweis. "Benefits and Impact of Cloud Computing on 5G Signal Processing: Flexible centralization through cloud-RAN". En: *IEEE Signal Processing Magazine 31.6* (Noviembre 2014), págs 35-44. ISBN: 1053-5888. DOI: 10.1109/MSP.2014.2334952.

[22] P. Arnold, N. Bayer, J. Belschner y G. Zimmermann. "5g radio access network architecture based on flexible functional control / user plane splits". En: *2017 European Conference on Networks and Communications (EuCNC)* (Junio 2017), págs 1-5. DOI: 10.1109/EuCNC.2017.7980777.

[23] D. Harutyunyan y R. Riggio. "Flex5g: Flexible functional split in 5g networks". En: *IEEE Transactions on Network and Service Management, vol. 15, no. 3* (Septiembre 2018), págs 961–975. DOI: 10.1109/TNSM.2018.2853707.

[24] A. Checko, A. P. Avramova, M. S. Berger y H. L. Christiansen. "Evaluating C-RAN fronthaul functional splits in terms of network level energy and cost savings". En: *Journal of Communications and Networks, vol. 18, no. 2* (2016), págs 162–172. DOI: 10.1109/JCN.2016.000025.

[25] Klaus-D. Hackbarth. "Introducción a la Planificación, Diseño, Dimensionado de redes de telecomunicación". En: *Planificación, Diseño y Dimensionado de Redes Móviles*. URL: https://www.tlmat.unican.es/siteadmin/submaterials/231.pdf (visitado 11-06-2020).

[26] Ramón Agüero. "Teletráfico. Dimensionado de Sistemas". En: *Redes de Comunicaciones*. URL: https://www.tlmat.unican.es/siteadmin/submaterials/3186.pdf (visitado 11-06-2020).

[27] Ramón Agüero. "Modelo M/M/1 y extensiones". En: *Dimensionado y Planificación de Redes*. URL: https://www.tlmat.unican.es/siteadmin/submaterials/3028.pdf (visitado 11-06-2020).

[28] 3GPP. "Study on new radio access technology: Radio access architecture and interfaces". En: *3rd Generation Partnership Project (3GPP), TR 38.801, 2017*. URL: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3056 (visitado 15-06-2020).

[29] N. J. Gomes, P. Sehier, H. Thomas, P. Chanclou, B. Li, D. Munch, P. Assimakopoulos, S. Dixit, y V. Jungnickel. "Boosting 5G Through Ethernet: How Evolved Fronthaul Can Take Next-Generation Mobile to the Next Level". En: *IEEE Vehicular Technology Magazine, vol. 13, no. 1* (2018), págs 74–84. DOI: 10.1109/MVT.2017.27823585.

[30] Luis Diez, Cristina Hervella, Ramón Agüero. "A Quasi-Birth-Death model for Functional Split in 5G Controllers". Enviado a: *Global Communications Conference. GLOBECOM 2020* (Junio 2020), págs 1–6.

[31] Luis Diez, Cristina Hervella, Ramón Agüero. "Understanding the performance of flexible functional split in 5G vRAN Controllers: A Markov Chain-based model". Enviado a: *IEEE Transactions on Network and Service Management* (Junio 2020), págs 1–12.